LA-UR-13-28028
Approved for public release; distribution is unlimited.

Title: 2013 Final Reports from the Los Alamos National Laboratory Computational Physics Student Summer Workshop

Author(s): Runnels, Scott R.

Intended for: Report
Web

Issued: 2013-10-16

2013
Final Reports

From the

Los Alamos National Laboratory Computational Physics Student Summer Workshop

Assembled by: Scott R. Runnels, Ph.D.
Workshop Coordinator and
University Liaison for LANL's Advanced
Scientific Computing Program

Included in this Report

Table of Contents

Diffusion in Mixed Cells (William Dai, mentor)

Turbulence Modeling (Dan Israel and John Schwarzkopf, mentors)

Background

Philosophy of the Workshop

The two primary purposes of LANL's Computational Physics Student Summer Workshop are (1) To educate graduate and exceptional undergraduate students in the challenges and applications of computational physics of interest to LANL, and (2) Entice their interest toward those challenges. Computational physics is emerging as a discipline in its own right, combining expertise in mathematics, physics, and computer science. The mathematical aspects focus on numerical methods for solving equations on the computer as well as developing test problems with analytical solutions. The physics aspects are very broad, ranging from low-temperature material modeling to extremely high temperature plasma physics, radiation transport and neutron transport. The computer science issues are concerned with matching numerical algorithms to emerging architectures and maintaining the quality of extremely large codes built to achieve multi-physics calculations. Although graduate programs associated with computational physics are emerging, it is apparent that the pool of U.S. citizens in this multi-disciplinary field is relatively small and is typically not focused on the aspects that are of primary interest to LANL. Furthermore, more structured foundations for LANL interaction with universities in computational physics is needed; currently interactions rely almost solely on individuals' personalities and personal contacts. Thus a tertiary purpose of the Summer Workshop is to build an educational network of LANL researchers, university professors, and emerging students to advance the field and LANL's involvement in it.

This was the third year for the Summer Workshop. Like the previous years, the workshop's goals were achieved by bringing into LANL a select group of students recruited from across the United States and immersing them for ten weeks in lectures and interesting research projects. The lectures provided an overview of the computational physics topics of interest this year along with some detailed instruction while the projects gave the students a positive experience accomplishing technical goals. Each team consisted of two students working under one or more mentors from LANL on specific research projects associated with predefined topics. This year, the topics were algorithms for shock hydrodynamics, plasma mixing in ICF applications, 3-T plasma physics, new algorithms for GPUs, MCNP Monte Carlo, X-ray Thomson scattering of warm dense matter, unstructured mesh algorithms for multi-core/GPU computers, verification problems for rad-hydro, material interfaces in rad-hydro, fission-fragment charge yield distribution calculations, diffusion in mixed cells, and turbulence modeling.

The students' growth was furthered by their participation on teams where their teammates were sometimes of different academic rank. It also developed their skills by requiring them to produce written and oral reports that they presented to peers, mentors, and management.

Funding and Participation Profile

LANL Staff

The Advanced Scientific Computing (ASC) Program at Los Alamos National Laboratory, under charge code JPDJ, sponsors the Summer Workshop by funding the workshop coordinator, paying for the lease at the University of New Mexico – Los Alamos campus, and also funding six of the twenty-four students. The remaining eighteen students were funded by various projects (some of them under ASC and some not), as shown below. This year, there were fifteen mentors, up from nine the previous year. The participation amongst different divisions grew again this year, including mentors from XCP, XTD, T, CCS, and HPC. This broad participation is welcome and it is hoped that it continues in future years. The details of the funding and divisional participation are summarized below.

Charge Code	Mentor	Home Division	Topic
JA1F	Brown	XCP	Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo
JADD	Dai	HPC	Diffusion in Mixed Cells
JA2K	Fung/Kenamond	XCP	The VEX Radiation Module: 2D Radiation Transport with Mimetic Diffusion for ExaFLAG
J444	Israel/Schwarzkopf	XCP/XTD	Turbulence modeling
JA2P	Masser	CCS	Improved Time Step Controller for xRage
JPDJ	Moller	T	Accelerating a Metropolois Random Walk and Immersion-Method Saddle-Point Algorithms
JPDJ	Morgan	XCP	One Dimensional Lagrangian Hydrocode Development
JA4B	Ramsey	XCP	Verification Problems for xRAGE Radiative Hydrodynamics Code
JA2J	Robey	XCP	New Algorithms for GPUs
JAUE	Saumon	XCP	Modeling X-ray Thomson scattering spectra of warm dense matter
JW61	Urbatsch/Runnels	XTD/XCP	Using High-Fidelity Radiation Transport Methods to Supplement the Diffusion Approximation at Material Interfaces
JPDJ	Vold	XCP	Plasma mixing in ICF applications

Students

Seventy-two students applied for admission to the workshop, all eligible U.S. citizens with the breakdown shown in the chart that follows. The twenty-four that ultimately were selected and participated were from the following schools: Arizona State, BYU, Stanford University, University of Minnesota, Virginia Tech, UC Santa Cruz, University of New Mexico, Columbia, University of Illinois Urbana-Champaign, University of Tennessee Knoxville, MIT, University of Wisconsin, University of Oregon, Princeton, Florida State, University of Michigan, Indiana University. Their rank breakdown is also shown in the chart below.

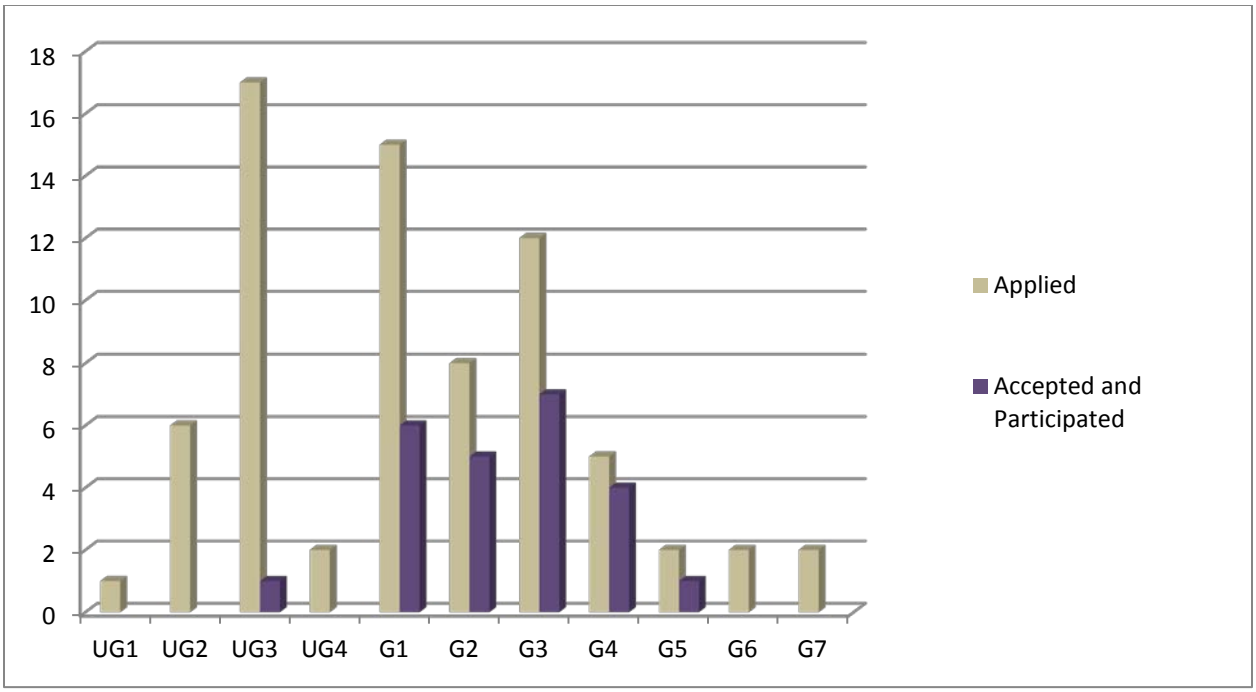


Chart showing the academic rank breakdown of the applicant pool and the ultimate participants. "UG1" means freshman, "UG2" means sophomore, etc., while "G1" means "1st year graduate student," "G2" means "2nd year graduate student," etc.

Lectures

The workshop coordinator and participating students greatly appreciated the contributions made by several lecturers, including some from outside LANL. The lectures were scheduled so the students could obtain the most benefit from them. Specifically, they were most frequent in the beginning of the workshop, when the students' research was just getting started and they needed the most background information. Then, their frequency dropped significantly until, finally, there were no lectures towards the end so the students could focus on their research without interruption.

The Summer Workshop strives to balance lectures with projects. With the growth in the number of mentors, students, and topics, it became necessary this year to introduce the concept of mandatory lectures versus optional lectures. The mandatory lectures are considered core to the program and part of the essential educational experience, while the optional lectures are designed more for the students focusing on that research topic. In 2012, there were approximately 42 hours of lectures provided, with the implicit understanding that they were all mandatory. This year, the mandatory lectures were reduced to 34 hours, but an additional 24 hours of optional, more in-depth lectures were given. In general, attendance at the optional lectures was strong.

The lectures are summarized on the next page.

Lectures – Required Attendance

Name	Affiliation	Topic	Length in hours
Scott Runnels	LANL	Intro. to Hydro Terminology	2
		Intro. to Grid Data Structures	1
Erik Vold	LANL	Basic Issues in Transport/CFD	1
		ICF Plasma Diffusion/Fluid Modeling	1
Nathaniel Morgan	LANL	Intro. to Lagrange Hydro	1
		Lagrange SGH in r-z	1
		Intro. to ALE	1
		Intro. to CCH Hydro	1
		Intro to FE PCH	1
Bob Robey	LANL	Parallel Programming	2
John Schwarzkopf	LANL	Turbulence	1
Dan Israel	LANL	Turbulence	1
Peter Moller	LANL	Essential Structures on Discrete Grids	1
Forrest Brown	LANL	Monte Carlo Foundations	3
		Basic Monte Carlo Techniques	2
William Dai	LANL	Diffusion/Mix Methods	1
Jim Kamm and Greg Weirs	Sandia	VVUQ	3
Malaya/Schultz	UT-Austin	Intro to Software Engineering	1
		Intro to MMS	1
Scott Ramsey	LANL	Test Problems	1
John Wallin	MTSU	Astrophysical Motions	1
		Simulation/Image Data Comparison	1
Mack Kenamond	LANL	Slidelines	1
Jimmy Fung	LANL	Mimetic Diffusion Solvers	1
Didier Saumon	LANL	Intro EOS Physics	1
Bill Rider	LANL	History/Codes	2

Lectures – Optional Attendance

Name	Affiliation	Topic	Length in hours
Scott Runnels	LANL	Intro. to Artificial Viscosity	1
Rob Cunningham	LANL	High-Performance Computing at LANL	2
Dan Israel	LANL	Turbulence Modeling	1
Peter Moller	LANL	Nuclear Masses/Fission Properties	1
		Nuclear Masses/Fission Properties	1
Forrest Brown	LANL	Basic Monte Carlo Techniques	1
		Monte Carlo Advanced Topics	1
		MCNP Tutorial	6
Brian Kiedrowski	LANL	Monte Carlo Advanced Topics	1
Chacon	LANL	PIC Algorithm	1
Nick Malaya and Karl Schultz	UT-Austin	MASA – A package for manufactured solutions verification	3
Scott Ramsey	LANL	Verification in Practice	1
Bill Rider	Sandia	Hydro Methods	2
Masser	LANL	PDE Verification	2

Survey Results

Twenty-two of the twenty-four workshop students participated in an anonymous survey during the final week of the workshop. They were asked to respond to the thirty-two assertions, below, using the response "Strongly Disagree", "Disagree", "Neutral", "Agree", or "Strongly Agree." In general, responses were very positive and appreciative. Detailed results are provided on the pages that follow.

Lectures

(1) Overall, the lecture quality was very good.
(2) There were some lectures that were exceptionally good.
(3) There were some lectures that need improvement.
(4) The overall scope of the lecture series was appropriate.
(4) I wished the series followed a more logical order.

Facilities

(1) I liked the room I was in.
(2) My chair was comfortable.
(3) My desk was comfortable.
(4) The noise level was acceptable.
(5) The temperature was about right.
(6) Access to the facilities was good enough.

Computing

(1) Moonlight was effective for me.
(2) The Sunrays were effective for me.
(3) I felt it easier just to use my own laptop.
(4) I used a combination of my laptop and LANL's computers.

LANL

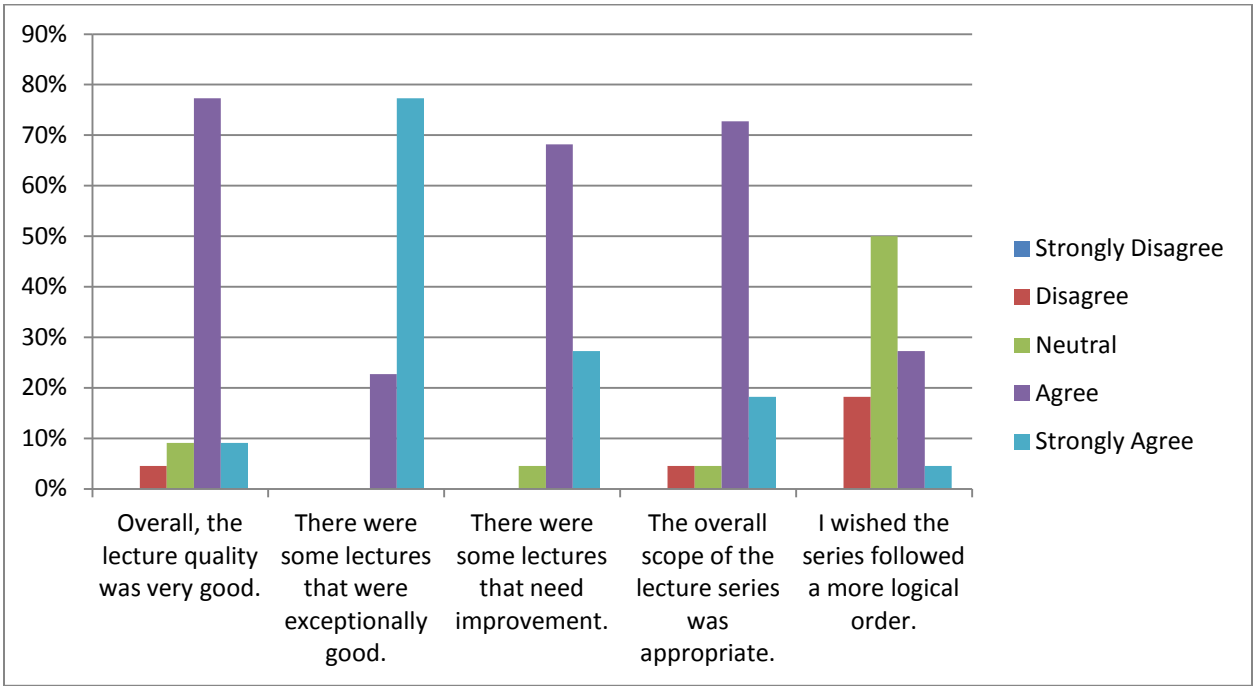
(1) The workshop gave me a positive impression of LANL.
(2) I felt the workshop was indicative of life at LANL.
(3) I think LANL would be a good place to work.

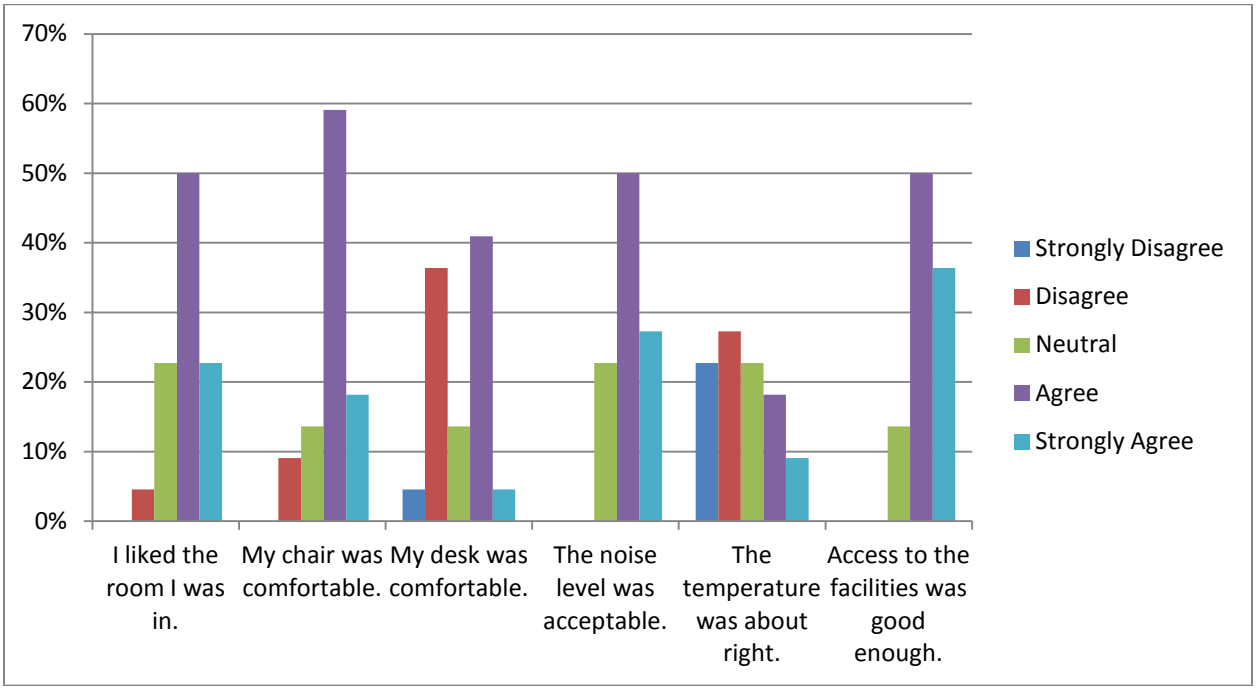
Mentors

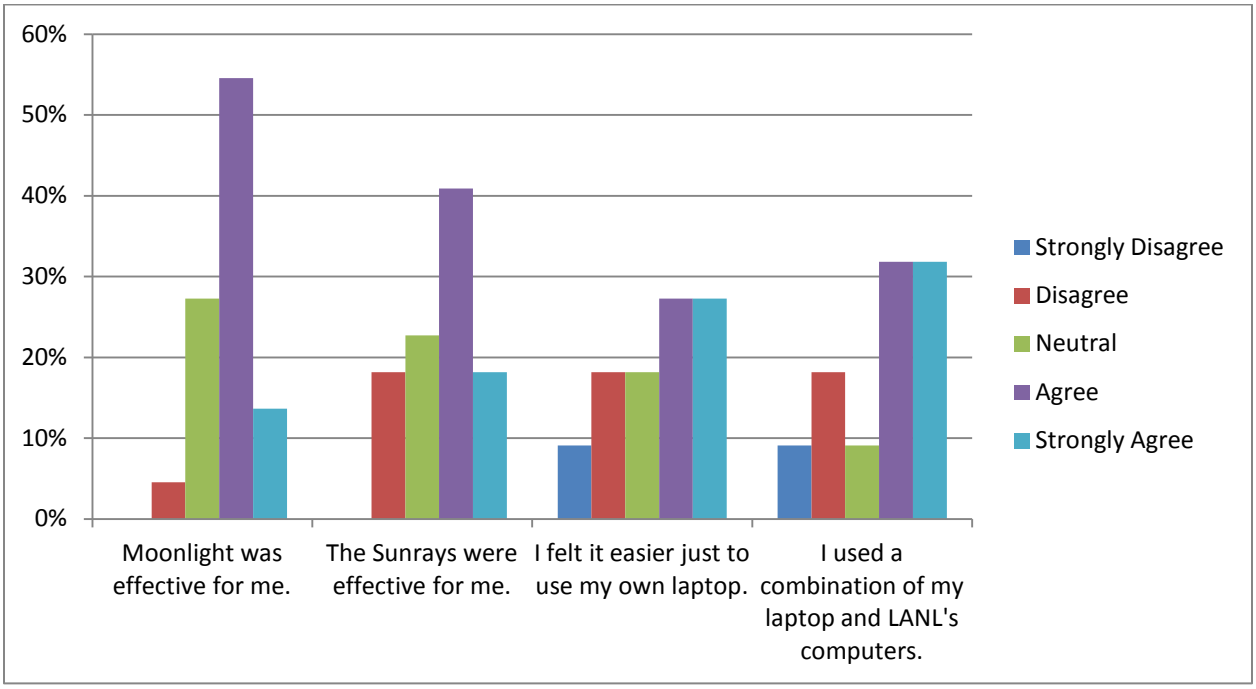
(1) My mentor spent enough time with me.
(2) My mentor did a good job in scoping out my project.
(3) My mentor worked hard to make me successful.
(4) My mentor was knowledgeable in the technical area.
(5) My mentor treated me with respect.
(6) My mentor served as a good example for me to follow.

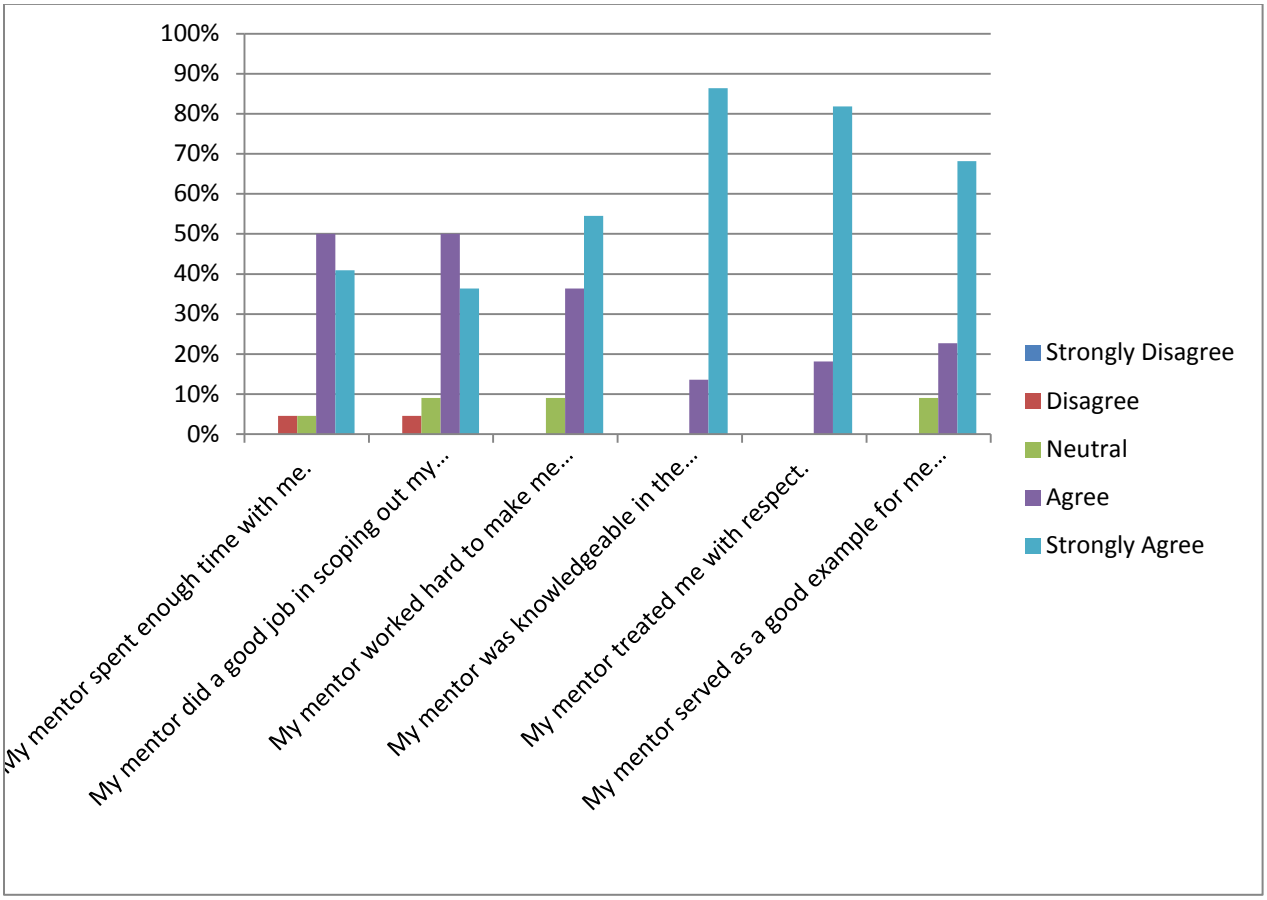
Organizer

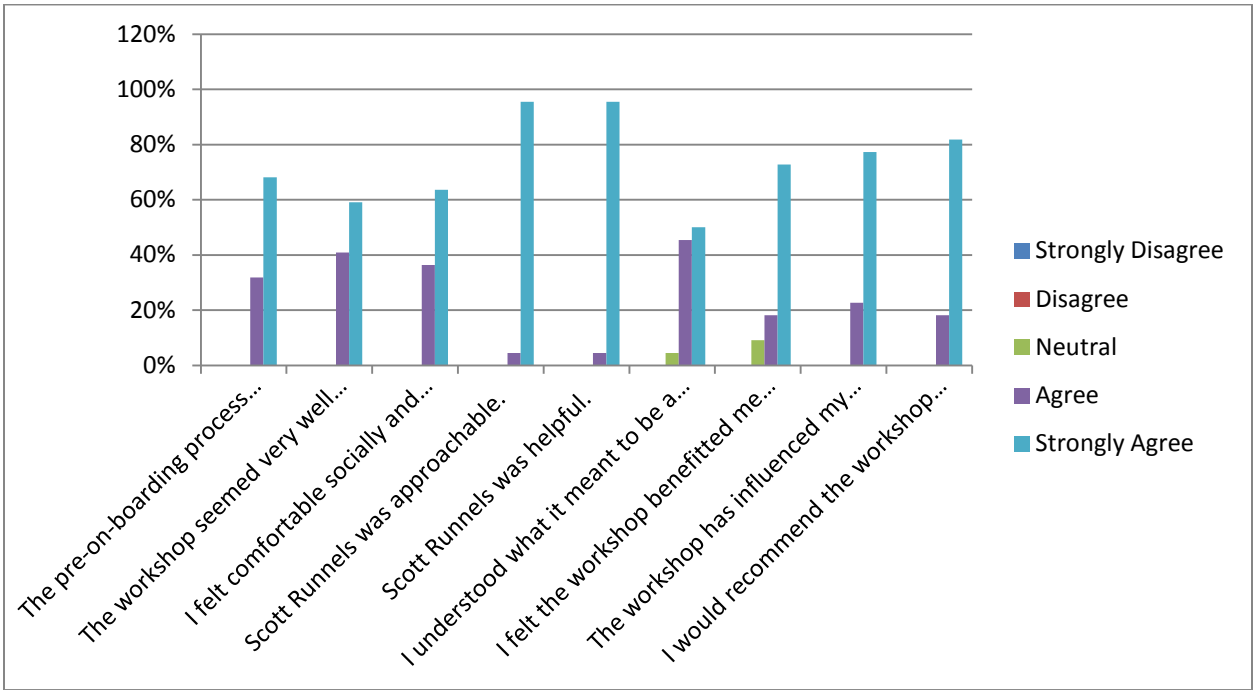
(1) The pre-on-boarding process worked really well for me.
(2) The workshop seemed very well organized.
(3) I felt comfortable socially and professionally.
(4) Scott Runnels was approachable.
(5) Scott Runnels was helpful.
(6) I understood what it meant to be a fellowship student.
(7) I felt the workshop benefitted me academically.
(8) The workshop has influenced my career in a positive way.
(9) I would recommend the workshop to a friend.

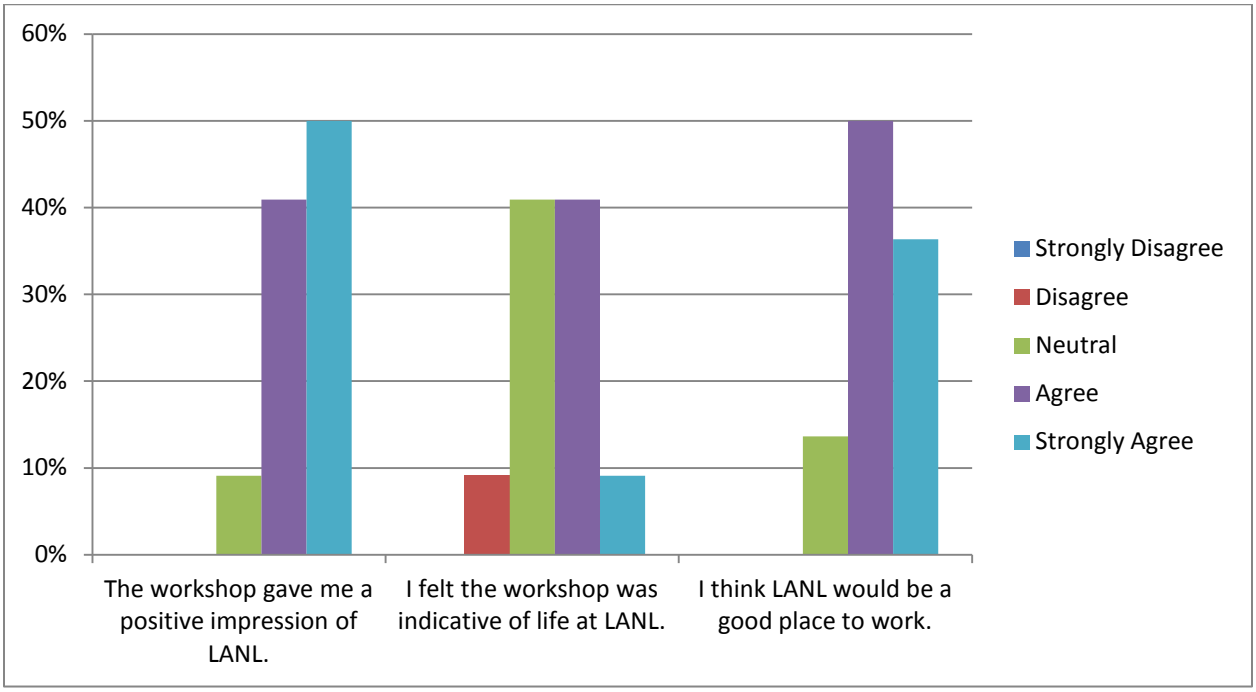

90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Overall, the lecture quality was very good.
There were some lectures that were exceptionally good.
There were some lectures that need improvement.
The overall scope of the lecture series was appropriate.
I wished the series followed a more logical order.
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


70%
60%
50%
40%
30%
20%
10%
0%
I liked the room I was in.
My chair was comfortable.
My desk was comfortable.
The noise level was acceptable.
The temperature was about right.
Access to the facilities was good enough.
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


60%
50%
40%
30%
20%
10%
0%
Moonlight was effective for me.
The Sunrays were effective for me.
I felt it easier just to use my own laptop.
I used a combination of my laptop and LANL's computers.
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
My mentor spent enough time with me.
My mentor did a good job in scoping out my...
My mentor worked hard to make me...
My mentor was knowledgeable in the...
My mentor treated me with respect.
My mentor served as a good example for me...
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


120%
100%
80%
60%
40%
20%
0%
The pre-on-boarding process...
The workshop seemed very well...
I felt comfortable socially and...
Scott Runnels was approachable.
Scott Runnels was helpful.
I understood what it meant to be a...
I felt the workshop benefitted me...
The workshop has influenced my...
I would recommend the workshop...
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree


60%
50%
40%
30%
20%
10%
0%
The workshop gave me a positive impression of LANL.
I felt the workshop was indicative of life at LANL.
I think LANL would be a good place to work.
Strongly Disagree
Disagree
Neutral
Agree
Strongly Agree

Student Reports

The reports that follow are assembled from separate PDF files. The table of contents at the beginning of this document uses page numbers in this fully assembled PDF file. In other words, it is recommended that the reader use the page indicator in the PDF viewer as the page number when navigating this combined document.

One Dimensional Lagrangian Hydrocode Development

(Nathaniel Morgan, mentor)

LA-UR-13-26506
Approved for public release; distribution is unlimited.

Title: One Dimensional Lagrangian Hydrocode Development

Author(s): Esmond, Micah J.
Thurber, Andrew J.

Intended for: 2013 Computational Physics Summer Student Workshop,
2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)

Issued: 2013-08-16

2013 Computational Physics Workshop
Los Alamos National Laboratory

One Dimensional Lagrangian Hydrocode Development

Micah Esmond
Department of Mechanical Engineering
Virginia Tech
jemicah2@vt.edu

Andrew Thurber
Department of Mechanical Engineering
Virginia Tech
ajthurbe@vt.edu

Mentor:
Dr. Nathaniel Morgan
XCP-8: Verification and Analysis
Los Alamos National Laboratory
nmorgan@lanl.gov

August 16th, 2013

Table of Contents

Introduction

Arbitrary Lagrangian Eulerian (ALE) codes combine the Lagrangian and Eulerian techniques. The Lagrangian approach is used to evolve the mesh, and the Eulerian approach is used to remap physical quantities after the mesh has been relaxed. This project focuses on the development of a 1D Lagrangian code as a testbed for ALE techniques; various hydrodynamic methods can be compared within a single framework. Remapping and ALE have not yet been implemented, and the code remains fully Lagrangian. The development and use of this code will aid in the understanding of the mathematics behind these types of algorithms.

Code Details

The code requires user inputs stored in a text file that is called by the input function in the code. Having a separate text file containing user inputs allows for the inputs to be changed easily and eliminates the need to compile the code after a slight change in the inputs. The use of a separate text file for inputs also requires the development of a parser. The parser used in this code was developed with the help of Dr. Vincent Chiravalle. The parser reads in the input text file and separates each word, variable, and value and then stores them in an array of strings. The input function then queries the array and extracts user-defined values and assigns them to the simulation parameters. These simulation parameters include mesh density, domain length, boundary conditions, thermodynamic quantities, and output options.

Once the simulation parameters defined by the user are collected, the initialize function in the code uses them to populate 1D arrays that will be used in the simulation. The code utilizes these arrays to represent the spatial distributions of thermodynamic quantities. The arrays can represent multiple materials. For the simulations of interest in this project, the property that differentiates materials is the ratio of the specific heats, i.e. gamma. Each element in an array represents an individual control volume containing a single material. Once ALE is implemented in the code, individual elements will have to represent multiple materials because of remapping procedures. This will be done using dynamic link lists.

The physics of the simulations are carried out in separate functions called “CCH”, “PCH”, “MPC”, “SGH”. These functions contain the Runge-Kutta time integration steps which include the majority of the calculations involving the conservation equations and the equation of state. At the end of these functions, the maximum speed of sound in the computational domain is computed. This value is fed back into the main function to compute the next time step. The main function serves to calculate the next time step and print out current simulation information including the current time, time step, and the minimum Δx across the domain.

At user-specified time intervals, the output function records the spatial distributions of thermodynamic quantities in data files. The data files are named using the name of the input file with the time information appended to it. Output files are generated for both point and cell arrays. These data files can then be plotted using third party plotting software such as GNUplot.

Source Code Files

The source code files are presented and described in Table 1.

Table 1. Source code file names and descriptions.

Source code file name	Description
program.c	Contains the main function as well as optimization functions used for the computations. The main function calls the subsequent functions and organizes their respective inputs and outputs. The main function also calculates the next time steps to input into the hydro packages.
header.h	Spacial arrays and other simulation parameters are declared in this file in order to be used globally. This file is included in each of the source code files.
setDefaults.c	Contains the function that establishes simulation parameters to be used if the user does not include them in the input file. It provides additional robustness to minimize user frustration.
input.c	Contains the function that parses the user input file and saves to variables for further use. If inputs are omitted, the default values are used.
initialize.c	Contains the function allocates memory and defines arrays and variables included in header.h, based on values from the user input file.
CCH.c	Contains the cell-centered hydrodynamics function. This funtion calls a separate function that performs a Riemann solution and steps the simulation parameters forward in time. Separate volume and area have also been included. See CCH Description.
PCH.c	Contains the point-centered hydrodynamics function. This function is used for the PCH and PCHA methods. It includes various functions for the Riemann solution and the time integration. It also contains a functions that compute a volumes and areas. See PCH Description and PCH Method Comparison.
SGH.c	Contains the staggered-grid hydrodynamics code. There is also a separate function that performs a Riemann solution and steps the simulation parameters forward in time. Volumes and areas are also computed in separate functions. See SGH Description.
MPC.c	Contains the modified point-centered hydrodynamics code. This function also calls a separate function that performs a Riemann solution. Volumes and areas are also computed in separate functions.
output.c	Contains a function that generates output files at specific time intervals defined by the user. This time interval is referred to as *dtdump*. Values for relevant properties such as density, pressure, etc. are output at each point or cell location, depending on the hydro method selected by the user.

Each source code file includes the header.h file. When the source code is compiled using the makefile, an executable file named *code* is created. The executable file is run with an input file using the following syntax:

```
./code ExampleInput.inp
```

The flowchart in Figure 1 details the overall procedure of the 1D hydrocode.

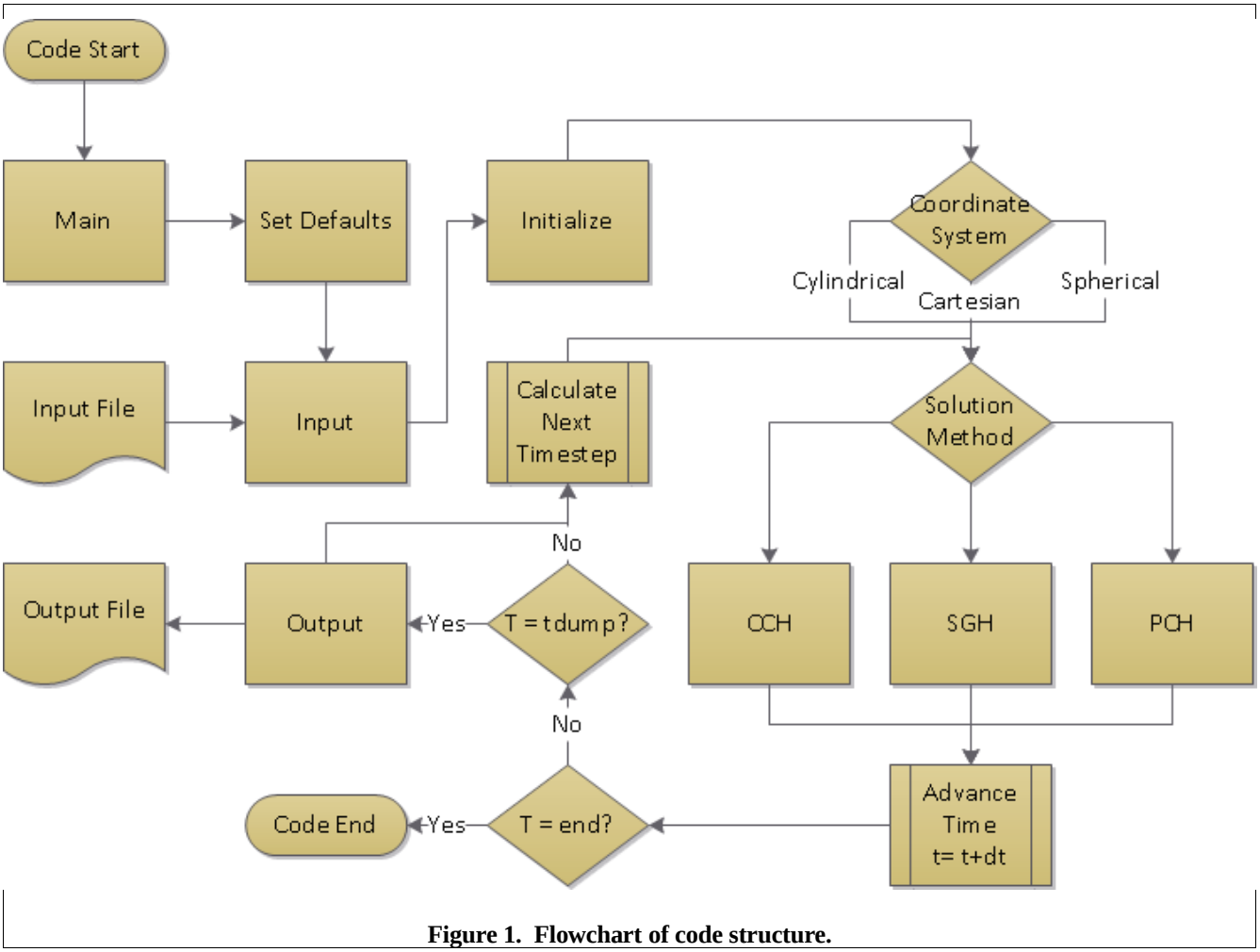


Figure 1. Flowchart of code structure.

Input

The user provides inputs to the hydrocode using a separate text file. If the user omits any inputs, default values will be used in their place. A list of the available user inputs along with the default values is shown in Table 2.

Table 2. User inputs and default values.

Input	Default Value	Comments
General Inputs		
mats	2	Number of Materials
np	20	Number of points used in the simulation
L	10	Length of domain (this is also the radius in curvilinear coordinates)
init-ie?	0	Initializes the simulation using specified internal energy (1) or pressure (0)
init_from_cell	0	Initializes point values based on the user inputs (0) or on cell values (1). Use (1) to smooth the shock front.
BC1	0	Fixed (0), Free (1)
BC2	0	Fixed (0), Free (1)
BCu1	0	Fixed wall velocity
BCu2	0	Fixed wall velocity
Method	CCH	
VelOpt	0	Computes density and total energy change from averaged point velocities (0) or from Riemann velocities (1)
AvgOpt	0	Averages the nodal velocities in space (0) or in space and time (1)
NodePosOpt	0	Updates the nodal positions based on the nodal velocities (0) or the averaged control volume boundary positions (1)
Coordinate_System	car	Planar (car), Cylindrical (cyl), Spherical (sph)
Material Inputs		
u	0	Initial velocity of each material
p	1.0 and 0.1	Initial pressures of material 1 and material 2, respectively
rho	1.0 and 0.1	Initial densities of material 1 and material 2, respectively
ie	1.0 and 0.1	Initial specific internal energies of material 1 and material 2, respectively
x1	0.0 and 5.0	Start locations of material 1 and material 2, respectively
x2	5.0 and 10.0	End locations of material 1 and material 2, respectively
gamma	1.4	Gamma of each material
I/O Parameters		
dt0	1e-9	Initial time step
tstop	30.0	Stop time
dtdump	5.0	Time interval to write output files
CFL	0.5	CFL parameter for time step control
CFLV	0.01	CFLV parameter for time step control

It is important to note that the materials must be entered in the order they appear in the domain from left to right. The 1D hydrocode assumes a consistent set of units. A typical system of units used for these problems is cm (length), μs (time), megabar (pressure), cc (volume), g (mass), megabar cc/g (specific energy).

CCH Description

Cartesian Coordinates

In CCH, all parameter values are stored at the cell's center. A Riemann-like solution is used to determine the velocity and pressure at the interface between cells, called points or nodes. See Figure 2.

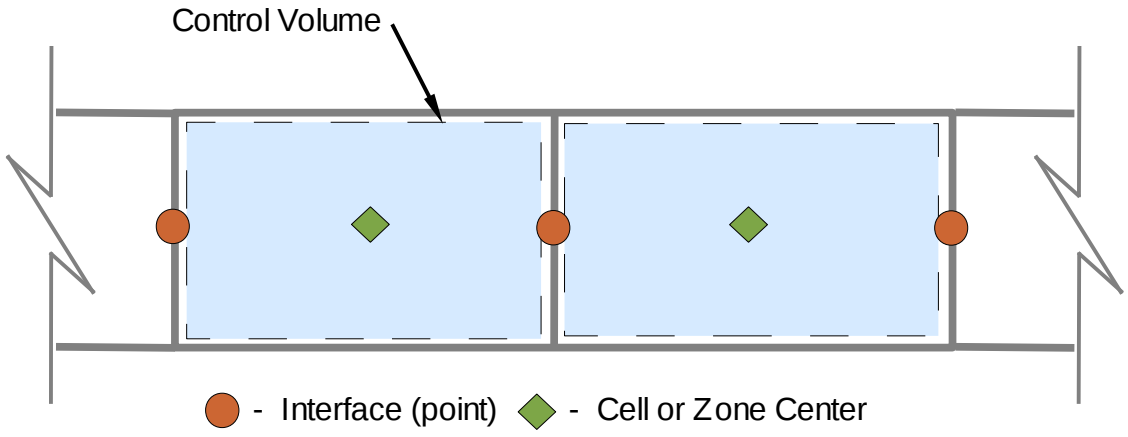


Figure 2. CCH mesh.

For 1D, the CCH method uses the following governing equations. The continuous equations are on the right, and the discretized approximations are shown on the right.

Continuous	**1D Discrete**
$\frac{d}{dt}\int_V \rho dV = 0$	$\frac{\Delta M_z}{\Delta t} = 0$
$\bigtriangledown \cdot \mathbf{u} = \lim_{V\to 0} \frac{1}{V}\frac{\delta V}{\delta t}$	$\frac{\Delta V_z}{\Delta t} = \sum_{CV} (\mathbf{n}A\ u^*)^{n+\frac{1}{2}}$
$\frac{d}{dt}\int_V \rho \mathbf{u} dV = \oint_A \mathbf{n}\cdot(-P\mathbf{I})dA$	$M\frac{\Delta u_z}{\Delta t} = -\sum_{CV} (\mathbf{n}A\ P^*)^{n+\frac{1}{2}}$
$\frac{d}{dt}\int_V \rho\left(e+\frac{u^2}{2}\right)dV = \oint_A \mathbf{n}\cdot(-P\mathbf{I})\cdot\mathbf{u}\,dA$	$M\frac{\Delta j_z}{\Delta t} = -\sum_{CV} (\mathbf{n}A\ P^*\ u^*)^{n+\frac{1}{2}}$

where *M* is the mass, *V* is the volume, **u** is the velocity, *P* is the pressure, *j* is the specific total energy, and **n** is the surface normal vector that points in either the positive or negative direction in 1D. The superscript * indicates the Riemann solution which is discussed in a later section. The subscript *z* indicates a zone or cell centered quantity. The superscript *n+1/2* indicates the time integration scheme.

The time integration scheme is discussed in more detail in a later section. In 1D, the change in the x-location of the points is used to compute the change in volume. The change in the x-location of a point is determined by

$$\frac{\Delta x_p}{\Delta t} = u^{*(n+\frac{1}{2})}$$

The density is updated using the volume change. The internal energy is determined using the equation

$$e_z = j_z - \frac{1}{2}u_z^2$$

where e_z is the specific internal energy in the zone. Using the updated density and the internal energy, the pressure is updated using the equation of state for a gamma-law gas. A constant gamma is assumed. *P* is given by

$$P_z = \rho_z(\gamma - 1)e_z$$

Cylindrical Coordinates

For cylindrical coordinates in 1D, the same governing equations are used as in Cartesian coordinates. The difference is in the volume and area calculations. The volume is computed as the volume per radian and is given by

$$V_z = \frac{1}{2}(r_{p+1}^2 - r_p^2)$$

where *dz*, the depth of the cylinder, is understood to be of unit length. The area that is used to compute the forces on the individual control volumes is computed in order to preserve consistency with the divergence relationship. The divergence relationship is

$$\bigtriangledown \cdot \mathbf{u} = \lim_{V \to 0} \frac{1}{V}\frac{\delta V}{\delta t}$$

The right side can be expressed as

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{V_z}\left[\frac{\delta V_z}{\delta r_{p+1}}\frac{\delta r_{p+1}}{\delta t} + \frac{\delta V_z}{\delta r_p}\frac{\delta r_p}{\delta t}\right]$$

simplifying, one obtains

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}^2 - r_p^2)}\left[r_{p+1}u_{p+1} - r_p u_p\right]$$

Simplifying again yields

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}+r_p)}\frac{\left[r_{p+1}u_{p+1}-r_p u_p\right]}{r_{p+1}-r_p}$$

and

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}+r_p)}\frac{\delta r u}{\delta r} \approx \bigtriangledown \cdot \mathbf{u}$$

Therefore, using the average radius at a control volume to determine the area will be consistent with the divergence relationship. The area per radian used to determine the forces acting on the cell is given by

$$A = \frac{1}{2}(r_{p+1}+r_p)$$

where *dz* is understood to be of unit length.

Spherical Coordinates

For spherical coordinates in 1D, the same governing equations are used as in Cartesian coordinates. However, the volume and area calculations are different. The volume is computed as the volume per steradian and is given by

$$V_z = \frac{1}{3}(r_{p+1}^3 - r_p^3)$$

The area used to compute the forces acting on the individual control volumes is computed in order to preserve consistency with the divergence relationship presented in the previous section. Similar to the derivation in the previous section, it can be shown that

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^3 - r_p^3)}\left[r_{p+1}^2\, u_{p+1} - r_p^2\, u_p\right]$$

Simplifying yields

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)}\frac{\left[r_{p+1}^2\, u_{p+1} - r_p^2\, u_p\right]}{r_{p+1}-r_p}$$

and

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)}\frac{\delta r^2 u}{\delta r} \approx \bigtriangledown \cdot \mathbf{u}$$

From this derivation, the area per steradian is given by

$$A = \frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)$$

Boundary Conditions

Two different types of boundary conditions are available in the code. The first is a fixed or reflective boundary, the second is a free boundary condition. The details of the mathematics used to simulate these conditions are presented below

CCH: Fixed (Reflective) Boundary Condition

For a fixed boundary, the velocity at the boundary is set to a specific value set by the user. Figure 3 illustrates mesh at the boundary.

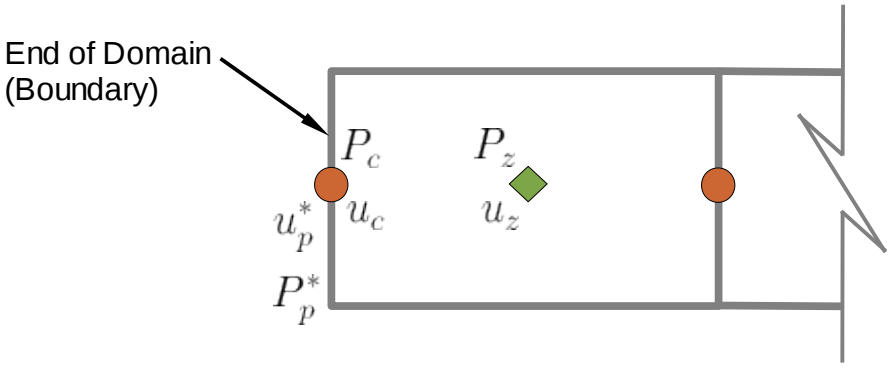


Figure 3. CCH boundary mesh.

The fixed velocity at the boundary is u^*_p. The pressure at the boundary is determined by the equation

$$P_p^* = P_c - \mu(u_p^* - u_c)\mathbf{n}$$

where **n** is the normal vector pointing in the positive or negative direction. μ at the boundary cell can be approximated by

$$\mu = \rho c$$

where the density and the speed of sound are evaluated at the boundary cell. Also, for a first order approximation,

$$u_c = u_z,\ P_c = P_z$$

Once the pressure and velocity at the boundary are known, the governing equations for the boundary

cell can be implemented normally.

CCH: Free Boundary Condition

For a free boundary condition, the pressure at the boundary must be maintained at zero. The velocity at the boundary must be determined. By manipulating the equation for *P** in the previous section, the following equation is obtained.

$$u_p^* = \frac{P_c \mathbf{n}}{\mu} + u_c$$

Where the approximations used for μ and the projected velocity and pressure are the same as for the fixed boundary condition.

SGH Description

In the SGH approach, the pressure, internal energy, and density are stored at the cell center. The velocity is stored at the points. The momentum and energy equations are solved on two separate control volumes. See Figure 4.

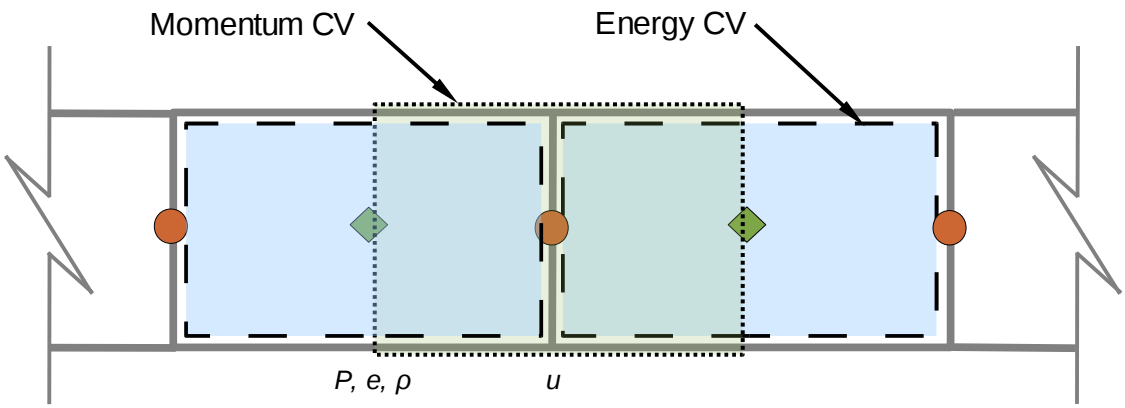


Figure 4. SGH mesh.

The momentum control volume is designated MCV, and the energy control volume is designated ECV. The discrete Lagrangian governing equations for SGH are shown below on the right and the continuous equations are shown on the left.

Continuous	1D Discrete
$\frac{d}{dt}\int_V \rho dV = 0$	$\frac{\Delta M_z}{\Delta t} = 0$
$\frac{d}{dt}\int_V \rho \mathbf{u} dV = \oint_A \mathbf{n}\cdot(-P\mathbf{I})dA$	$M_p \frac{\Delta u_p}{\Delta t} = -\sum_{MCV} (\mathbf{n}A\ P_z^*)^{n+\frac{1}{2}}$
$\frac{d}{dt}\int_V \rho e dV = (-P\mathbf{I}) : \oint_A \mathbf{n}\,\mathbf{u}\,dA$	$M_z \frac{\Delta e_z}{\Delta t} = -P_z^{*(n+\frac{1}{2})} \sum_{ECV} (\mathbf{n}A\ u_p)^{n+\frac{1}{2}}$

where M_p is the mass of the MCV centered at a point, and M_z is the mass of the ECV at a cell. e_z is the specific internal energy. The volume and density of the cell are updated based on the change in the locations of the points. The location of the points is updated using a time averaged velocity as in the equation.

$$\frac{\Delta x_p}{\Delta t} = \frac{1}{2}(u_p^n + u_p^{n+1}) \equiv u^{n+\frac{1}{2}}$$

A pressure is computed using the equation of state for a gamma-law gas where a constant gamma is assumed.

$$P_z = \rho_z(\gamma - 1)e_z$$

A Riemann-like problem is solved at the center of the each cell. In 1D, the velocities are projected from the points to the cell centers. The pressure stored at the cell center is the pressure used to determine u^* and P^*.

Curvilinear Coordinates

The equations implemented for curvilinear coordinates in SGH are derived similarly to those used for CCH. However, the area used in the momentum equation is evaluated at the center of the momentum control volume. This same area is then used in the energy equation. A predictor-corrector scheme was implemented in SGH. This procedure is explained in [1]. First, the nodal positions are estimated at the next time step using a time averaged velocity. The velocity is then updated using the areas computed from the predicted nodal positions. New nodal positions are then computed using the updated velocity and these values are compared to the predictions. The process is repeated until the difference in the predicted and computed nodal positions is negligible.

Boundary Conditions

SGH: Fixed (Reflective) Boundary Condition

The SGH approach utilizes a momentum control volume centered on the points and an energy control volume centered on the cells. To simulate a fixed boundary condition, the change in the velocity with respect to time for the boundary node is set to zero. To illustrate this concept, a customized momentum control volume is established for the boundary node as shown in Figure 5.

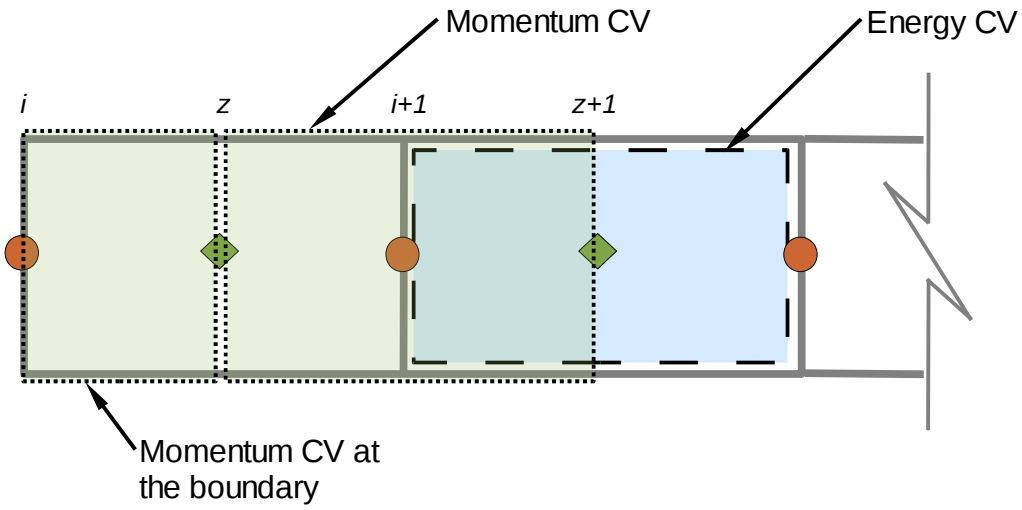


Figure 5. SGH boundary mesh.

For the momentum control volume at the boundary, the pressure on the right side must be equal to the pressure on the left side. Therefore, the change in velocity with respect to time for the boundary node must be zero to satisfy the governing equation

$$\frac{\Delta u_p}{\Delta t} = -\frac{1}{M_{MCV}} \sum_{CV} \mathbf{n}_i P_z^* A = 0$$

SGH: Free Boundary Condition

The free boundary condition for SGH is simulated by using the momentum control volume at the boundary and setting the pressure at the boundary node equal to zero. The rate of change of the velocity at the boundary node is then given by

$$\frac{\Delta u_p}{\Delta t} = -\frac{1}{M_{CV}} A\left[(P^*\mathbf{n})_i + (P^*\mathbf{n})_z\right] = -\frac{1}{M_{CV}} A(P_z^*\mathbf{n})_z$$

PCH Description

Using a point centered approach (PCH), pressure, energy, density, and velocity are stored at the points and each control volume is centered on a point. The Riemann-like solution is used to compute a *P** and *u** at the interfaces between the control volumes as shown in Figure 6.

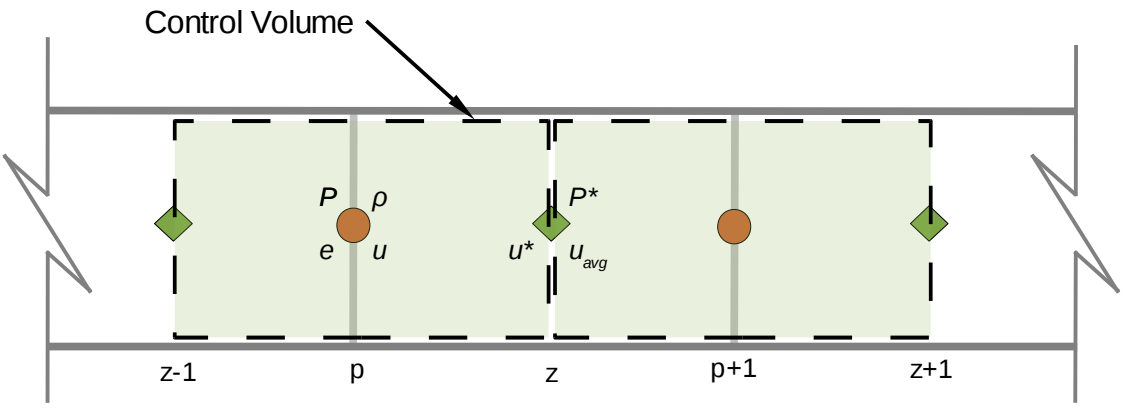


Figure 6. PCH mesh.

In 1D, the governing conservation equations reduce to following discretized forms:

$$\frac{\Delta M_p}{\Delta t} = 0$$

$$\frac{\Delta x_p}{\Delta t} = u_p^{n+\frac{1}{2}}$$

$$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^* u_{avg}$$

where M_p is the point centered mass and

$$F^* = (-\mathbf{n}A\ P^*)_z^{n+\frac{1}{2}}$$

$$u_{avg} = \frac{u_p^{n+\frac{1}{2}} + u_{p+1}^{n+\frac{1}{2}}}{2}$$

The internal energy is computed by subtracting the kinetic energy from the total energy, and the pressure is computed using the equation of state for a gamma-law gas assuming a constant gamma. For PCH, three different methods were used to update the density of each control volume. The governing equations presented in this section represent the first method used. Details regarding the other methods are presented in a later section.

Curvilinear Coordinates

The equations used to compute volumes and areas in PCH are similar to those used in CCH. The differences are that the volume is computed for the point centered control volumes rather than the cell centered control volumes, and that the areas are computed at the nodes rather than at the cell centers.

Boundary Conditions

At the boundaries, a control volume is not centered on a point, as it cannot extent past the domain, as seen below. The governing equations are then applied to a control volume that is typically half the size of an ordinary control volume in the domain. Figure 7 illustrates the PCH mesh at the boundary.

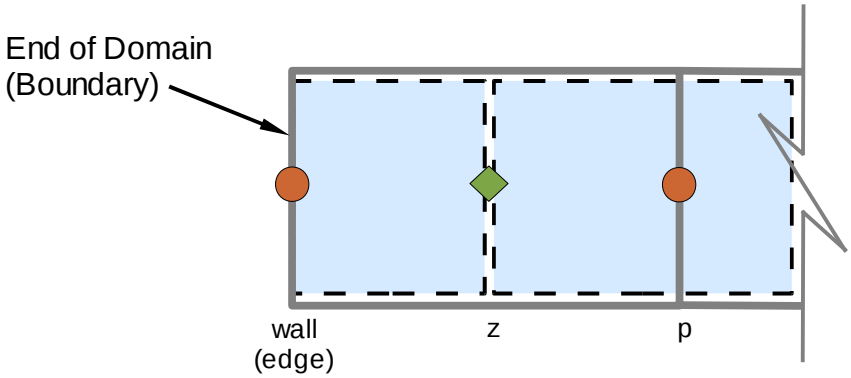


Figure 7. PCH boundary mesh.

PCH: Fixed BC

The fixed boundary is subject to the following conditions:

$$\left.\frac{\Delta x}{\Delta t}\right|_{wall} = u_{wall}$$

$$\left.\frac{\Delta u}{\Delta t}\right|_{wall} = 0$$

$$\left.\frac{\Delta j}{\Delta t}\right|_{wall} = \frac{1}{M_p}\sum_{CV} F^* u = \frac{1}{M_p}\left[\left(F^* u_{wall}\right)\big|_{wall} + \left(F^* u_{avg}\right)\big|_z\right]$$

where M_p is the mass of the control volume at the boundary and

$$u_{avg} = \frac{u_{wall} + u_p}{2}$$

$$F^*|_{wall} = -F^*|_z$$

The wall velocity, u_{wall}, is fixed. The magnitude of the forces on either side of the boundary control volume are assumed to be equal as the gradient of the pressure is set to zero for this condition.

PCH: Free BC

For a free boundary condition, the edge velocity is not fixed. The free boundary is subject to the following conditions:

$$\left.\frac{\Delta x}{\Delta t}\right|_{edge} = u|_{edge}$$

$$\left.\frac{\Delta u}{\Delta t}\right|_{edge} = \frac{1}{M_p}\sum_{CV} F^* = \frac{1}{M_p}F_z^*$$

$$\left.\frac{\Delta (te)}{\Delta t}\right|_{edge} = \frac{1}{M_p}\sum_{CV} F^* u = \frac{1}{M_p}\ F_z^* u_{avg}$$

where

$$u_{avg} = \frac{u_{edge} + u_p}{2}$$

The three different point centered methods discussed in this report vary slightly in the implementation of boundary conditions. For example, u_{avg} in the energy equation above is replaced by the zone-centered Riemann velocity, u^*.

Solver Details

Riemann Solution

As presented in [2], a linear relationship between the shock velocity and the material velocity (the *U-u* curve) can be a good approximation for many materials. This simplifies the Riemann solution. As pointed out in [3], the Riemann-like solution can be used to determine a viscous force acting on the material experiencing a shock. The Riemann-like solution solves the shock problem across the interface between two cells. This solution simulates the dissipation effects that characterize a shock. In 1D, the Riemann-like solution yields a velocity and pressure at the cell interface that are given by

$$u^* = \frac{(P_c^- \mathbf{n}^- + \mu^- u_c^-) + (P_c^+ \mathbf{n}^+ + \mu^+ u_c^+)}{(\mu^+ + \mu^-)}$$

$$-P^* \mathbf{n}^+ = \mu^+ (u^* - u_c^+) - P_c^+ \mathbf{n}^+$$

$$-P^* \mathbf{n}^- = \mu^- (u^* - u_c^-) - P_c^- \mathbf{n}^-$$

Where P_c and u_c are the projected values, and μ is the shock impedance. The + and – superscripts denote the positions of the quantities relative to the interface of interest. For simplicity, a first order projection is used for the velocity and pressure. In CCH, these values are projected from the zone center to the points. In SGH, the velocities are projected from the points to the zone center. The shock impedance is determined based on the linear approximation of *U*-*u* curve. For a gamma law gas, the slope of of the shock impedance relation can be approximated by $\left(\frac{\gamma+1}{2}\right)$ [4]. Therefore, the shock impedance is computed by

$$\mu^+ = \rho c + \rho \left(\frac{\gamma+1}{2}\right) \left\| \left(\frac{u_c^+ + u_c^-}{2} - u_c^+\right) \right\|$$

$$\mu^- = \rho c + \rho \left(\frac{\gamma+1}{2}\right) \left\| \left(\frac{u_c^+ + u_c^-}{2} - u_c^-\right) \right\|$$

where c is the acoustic wave speed. For a gamma-law gas, the acoustic wave speed is approximated by

$$c = \sqrt{\gamma \frac{P}{\rho}}$$

Time Integration

The code uses a fourth order, explicit Runge-Kutta time integration scheme. The Taylor series expansion of any function is

$$f^{n+1} = f^n + \Delta t \frac{\delta f}{\delta t} + \frac{\Delta t^2}{2!} \frac{\delta^2 f}{\delta t^2} + \frac{\Delta t^3}{3!} \frac{\delta^3 f}{\delta t^3} + \frac{\Delta t^4}{4!} \frac{\delta^4 f}{\delta t^4} + \ldots$$

If the higher order terms are neglected, the above equation can be represented by

$$f^{n+1} \approx f^n + \Delta t \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{2} \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{3} \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{4} \frac{\delta f}{\delta t} \right) \right) \right)$$

The code uses the quantities at “n” to compute a slope and step the quantities forward to “n+1/4”. The code then uses the quantities at “n+1/4” to compute a new slope and step the quantities forward from ”n” to “n+1/3”. This process is repeated until the quantities are stepped forward from “n” to “n+1”. At

each of these stages, a Riemann-like problem is solved at the control volume interfaces, and all the simulation parameters are stepped forward in time.

Time Step Control

Two methods are used in order to control the time step during the simulation. The first method is the Courant Stability Condition. The Courant stability condition requires that

$$\frac{u\Delta t}{\Delta x} \leq CFL$$

where *u* is the maximum speed of the fluid in the domain. The *CFL* parameter is chosen by the user. For explicit schemes, *CFL* must be less than or equal to 1. The maximum speed of sound, c_{max}, can be used in the place of the maximum speed of the fluid in the domain. c_{max} is used with the minimum Δx of a cell in the domain to calculate a new time step. The code calculates a new time step before each iteration by applying the formula

$$(\Delta t)_{new} = CFL\frac{(\Delta x)_{min}}{c_{max}}$$

The user can set an initial time step in the input file. The code allows the time step to grow by a maximum of 10% at each step.

The second method restricts the volume change of a cell in a single time step. The requirement is expressed by

$$CFLV \geq \frac{V^{n+1} - V^n}{V^n}$$

where *V* is the volume, and *CFLV* is a parameter set by the user. By restricting the volume change, other parameters, such as the density and pressure, are allowed to develop in the cell before the cell collapses and yields unphysical results. This adds stability to the computations. The fundamental formula is the divergence relationship.

$$\bigtriangledown \cdot \mathbf{u} = \lim_{V \to 0} \frac{1}{V}\frac{\delta V}{\delta t}$$

In 1D, the finite difference form of this equation can be expressed as

$$\frac{\Delta V}{\Delta t} = \frac{V^{n+1} - V^n}{\Delta t} = \frac{\Delta u}{\Delta x}V^n$$

By rearranging and substituting, the following formula gives a suitable time step.

$$\Delta t = \frac{CFLV}{\left(\left|\frac{\Delta u}{\Delta x}\right|^n\right)_{max}}$$

The hydrocode uses a CFLV parameter set by the user and finds the maximum velocity gradient in the domain. A new time step is then calculated using these values.

The time step used in the next calculation is the minimum of the time steps computed using these two methods.

PCH Method Comparison

The PCH method that was implemented in the code had difficulties on test problems with strong shocks, such as the Sod problem. As a results, a variety of PCH methods were explored. The first method is the canonical PCH method (PCH). In 1D, it was found that this method allowed the collapse of zones at shock discontinuities. The collapse of a cell causes the time step to approach zero and prevents the simulation from reaching the required time. It was found that by "smoothing" the shock interface initially eliminates the cell collapse issues. The smoothing was accomplished by averaging the thermodynamic quantities at the discontinuity. A second method seeks to solve the issue by updating the volume of each control volume using the averaged velocities at the control volume boundaries. This method is denoted by PCHA. This method allows the simulation to complete but does not update the nodal positions directly from the nodal velocities. Rather, the method updates the control volume boundaries and moves the nodal positions based on the new control volume boundaries. A third method uses the Riemann velocities at the control volume boundaries to calculate the density and total energy change. This method is denoted by MPC. This method is effectively identical to the CCH method, only shifted by half of a cell. The equations for each method are presented below.

PCH	**PCHA**	**MPC**
$\frac{\Delta M_p}{\Delta t} = 0$	$\frac{\Delta M_p}{\Delta t} = 0$	$\frac{\Delta M_p}{\Delta t} = 0$
$\frac{\Delta x_p}{\Delta t} = u_p^k$	$\frac{\Delta x_z}{\Delta t} = u_{avg}\big\|_z^k$	$\frac{\Delta x_z}{\Delta t} = u^*\big\|_z^k$
$V_p^{k+1} = \frac{x_{p+1}^{k+1} - x_{p-1}^{k+1}}{2}$	$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot (u_{avg})\big\|_z^k$	$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot u^*\big\|_z^k$
$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$	$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$	$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$
$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} (F^* u_{avg})\big\|_z^k$	$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} (F^* u_{avg})\big\|_z^k$	$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} (F^* u^*)\big\|_z^k$

where the area used to compute the volume is understood to be equal to 1 in planar coordinates. u_{avg} is the average of the neighboring nodal velocities. The procedures used to compute the volume change in PCH and PCHA are algebraically equivalent. The difference is that, in PCHA, the control volume boundary locations are updated using the averaged nodal velocities. These values are then used to

update the nodal positions after the time integration is complete. In PCH, the nodal positions are updated throughout the time integration using the nodal velocities.

Smoothing

In the Sod problem, the PCH method moves the individual points too quickly and causes the cell ahead of the shock discontinuity to collapse. This causes the time step to approach zero because the maximum allowable time step is based on the distance between individual points. One strategy to prevent this from happening is to smooth the shock discontinuity. This technique smooths out the shock interface by placing a point directly between the high and low density regions of Sod problem and assigns to the point the average of the two densities. The same technique is applied to the initial pressure and energy distributions. Figure 8 illustrates this technique.

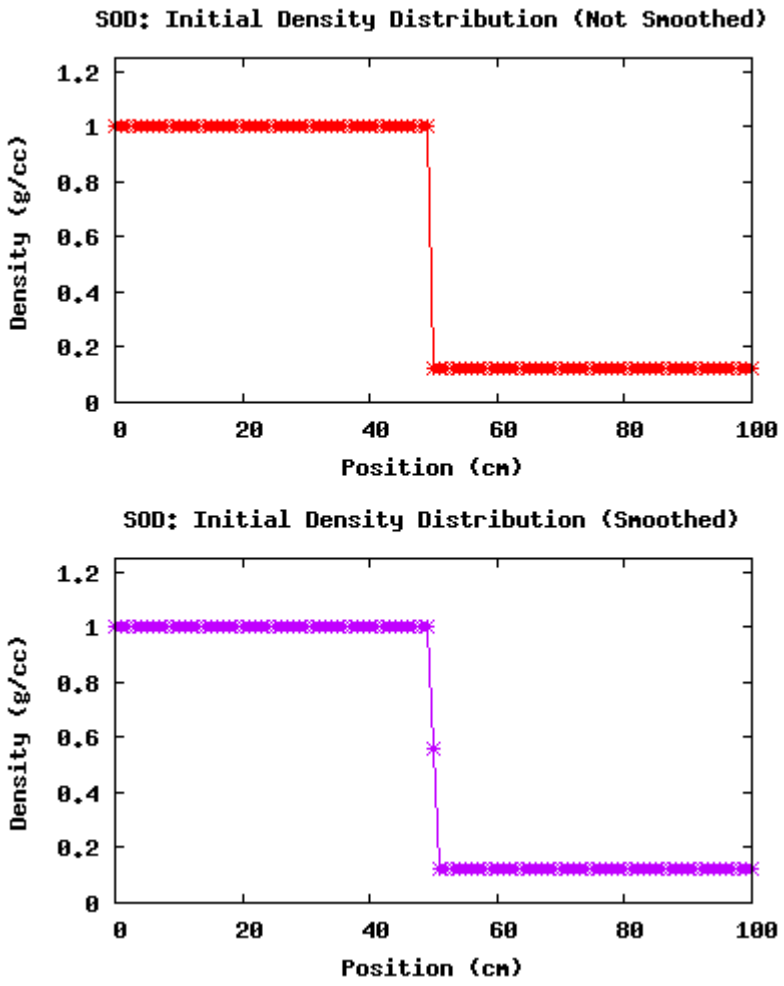


Figure 8. PCH: Smoothing technique concept.

This technique allows the simulation to run past 20 μs. The simulation results at 20 μs are compared to the analytical solution in Figure 9.

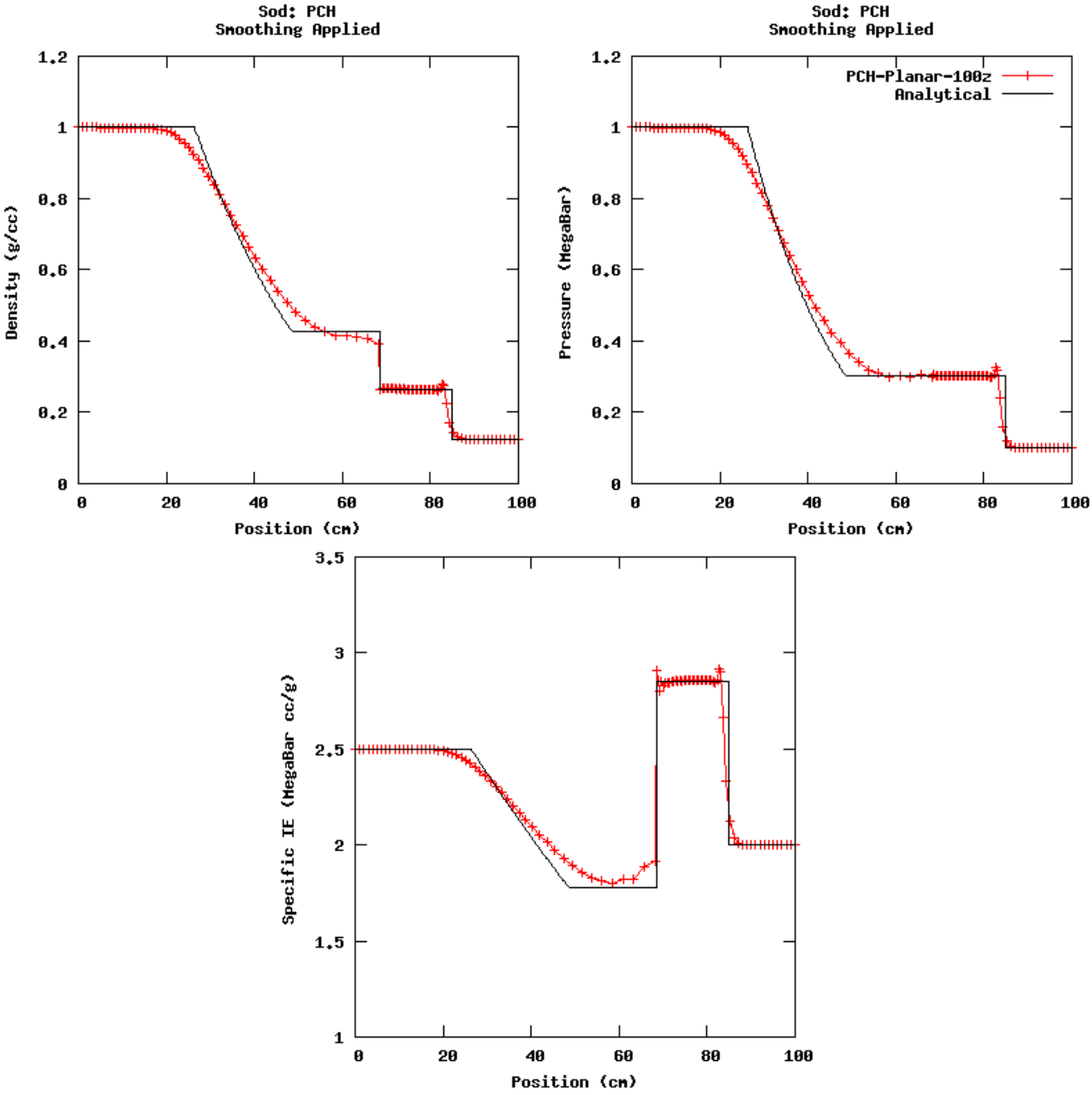


Figure 9. PCH Results for the Sod problem at 20 μs with smoothing applied.

The smoothing technique allows the simulation to run up to 41.72 μs. The results for pressure at this time are shown in Figure 10.

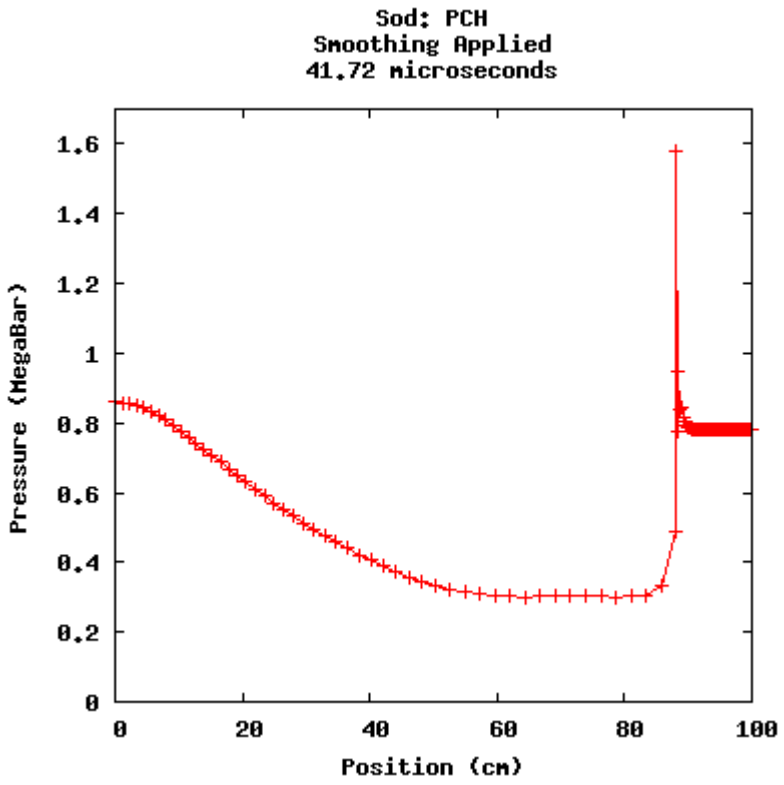


Figure 10. Maximum extent of PCH simulation with smoothing applied for the Sod problem.

At 41.72 μs, the shock has been reflected off of the boundary at 100 cm and has begun to move to the left. A new shock interface has formed, observed in the figure above at approximately 90 cm. This new interface has not been smoothed since no algorithm to automatically smooth shocks has been implemented. Therefore, the points move too close together and the time step approaches zero which stops the simulation.

Space and Time Averaged

For the PCHA method, the control volume boundaries are moved using the averaged nodal velocities. Two different approaches were investigated for this method. The first approach averages the nodal velocities in space, and the second approach averages the nodal velocity in space and time. For the second approach the governing equations for the control volume boundaries, volume, and total energy become

$$\frac{\Delta x_z}{\Delta t} = u_{avg}\Big|_z^{k+\frac{1}{2}}$$

$$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot u_{avg}\Big|_z^{k+\frac{1}{2}}$$

$$\frac{\Delta \dot{j}_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F_z^* u_{avg}\Big|_z^{k+\frac{1}{2}}$$

where

$$u_{avg}\Big|_z^{k+\frac{1}{2}} = \frac{1}{2}\left(\frac{u_p^n + u_p^{k+1}}{2} + \frac{u_{p+1}^n + u_{p+1}^{k+1}}{2}\right)$$

The different averaging approaches have little to no effect on the results. The results from a simulation using 100 zones and both averaging methods are compared to the analytical solution for the Sod problem at 20 μs in Figure 11.

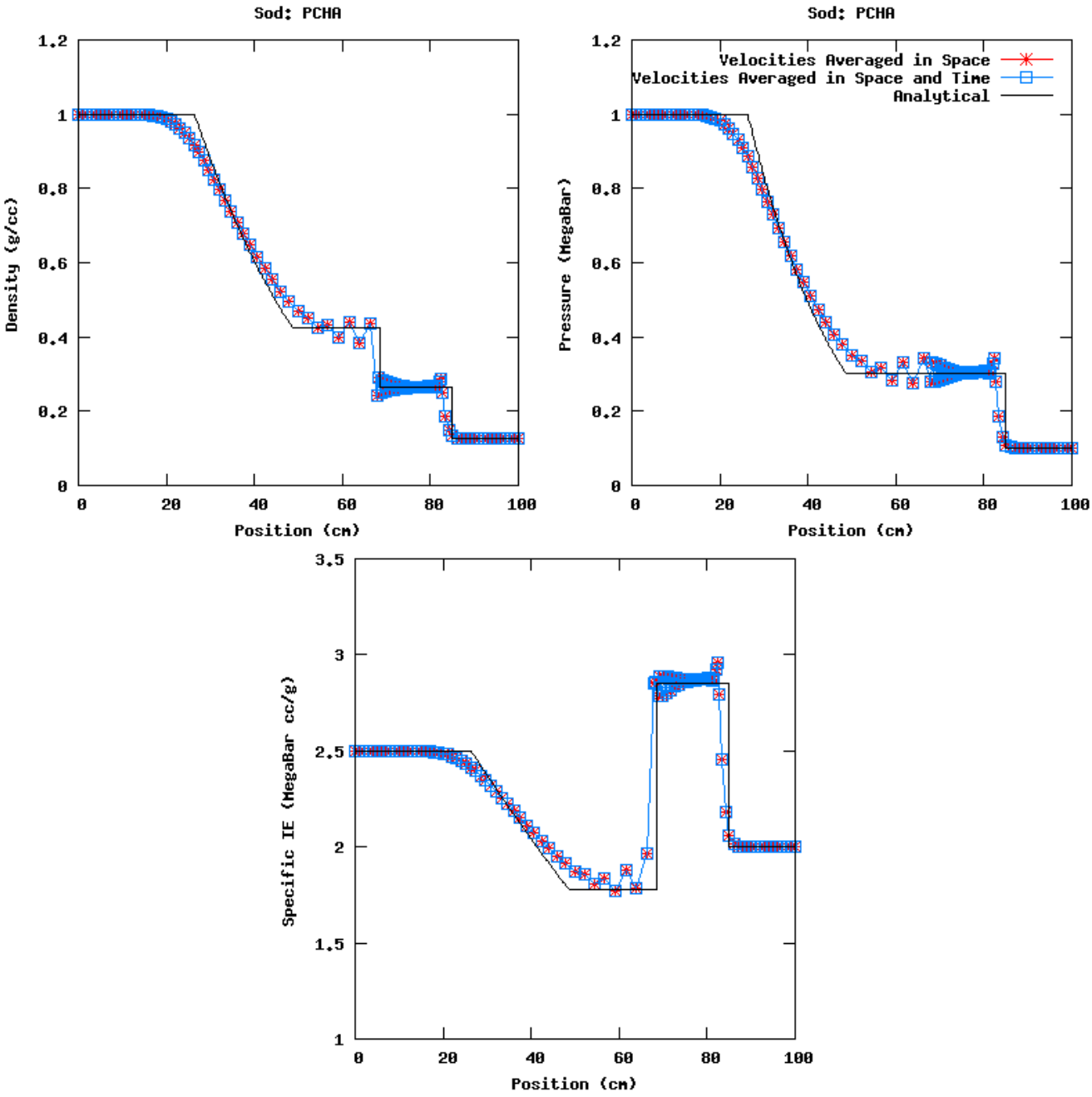


Figure 11. Comparison of results for the Sod problem. The PCHA simulation was run using the time averaged velocities as well as the time and spacial averaged velocities.

The approach used to calculate the average velocity, whether a space or a both space and time averaged velocity, does not change the results of the 1D simulation significantly. Even as the mesh is refined, there is no noticeable difference in the plotted results. For both approaches with 100 zones, a numerical ringing is produced at the contact discontinuity, located between 50 and 75 cm in Figure 11. The PCHA method used in the remainder of this report utilizes spatially averaged velocities. The effect of mesh refinement on the ringing is shown in Figure 12.

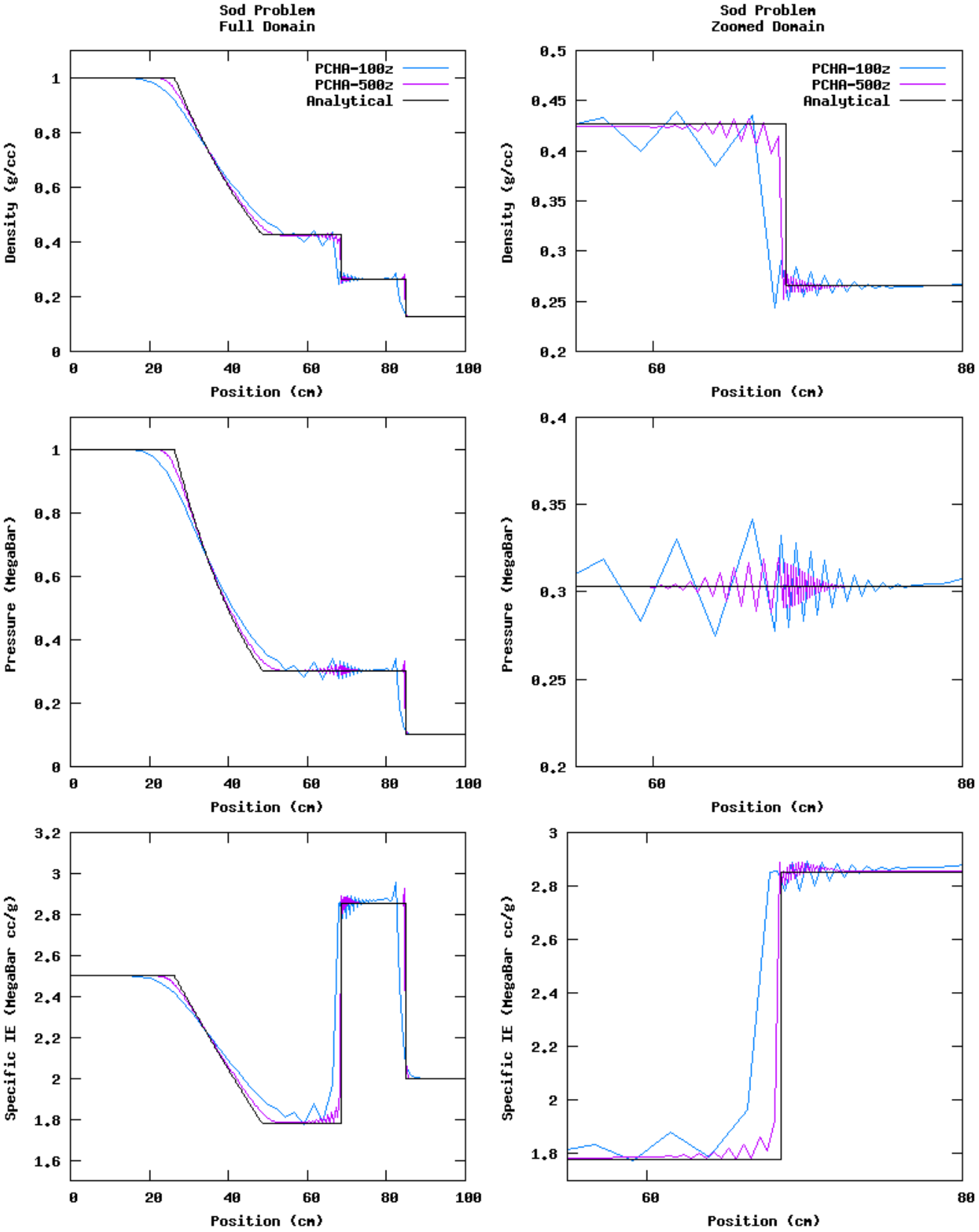


Figure 12. Effect of mesh refinement on the numerical ringing observed in the PCHA method.

PCH Comparison to PCHA

The PCH method experiences cell collapse for the Sod and Sedov test problems. However, for the Piston and Noh test problems, the PCH and PCHA methods produce similar results. These results are shown in Figure 13.

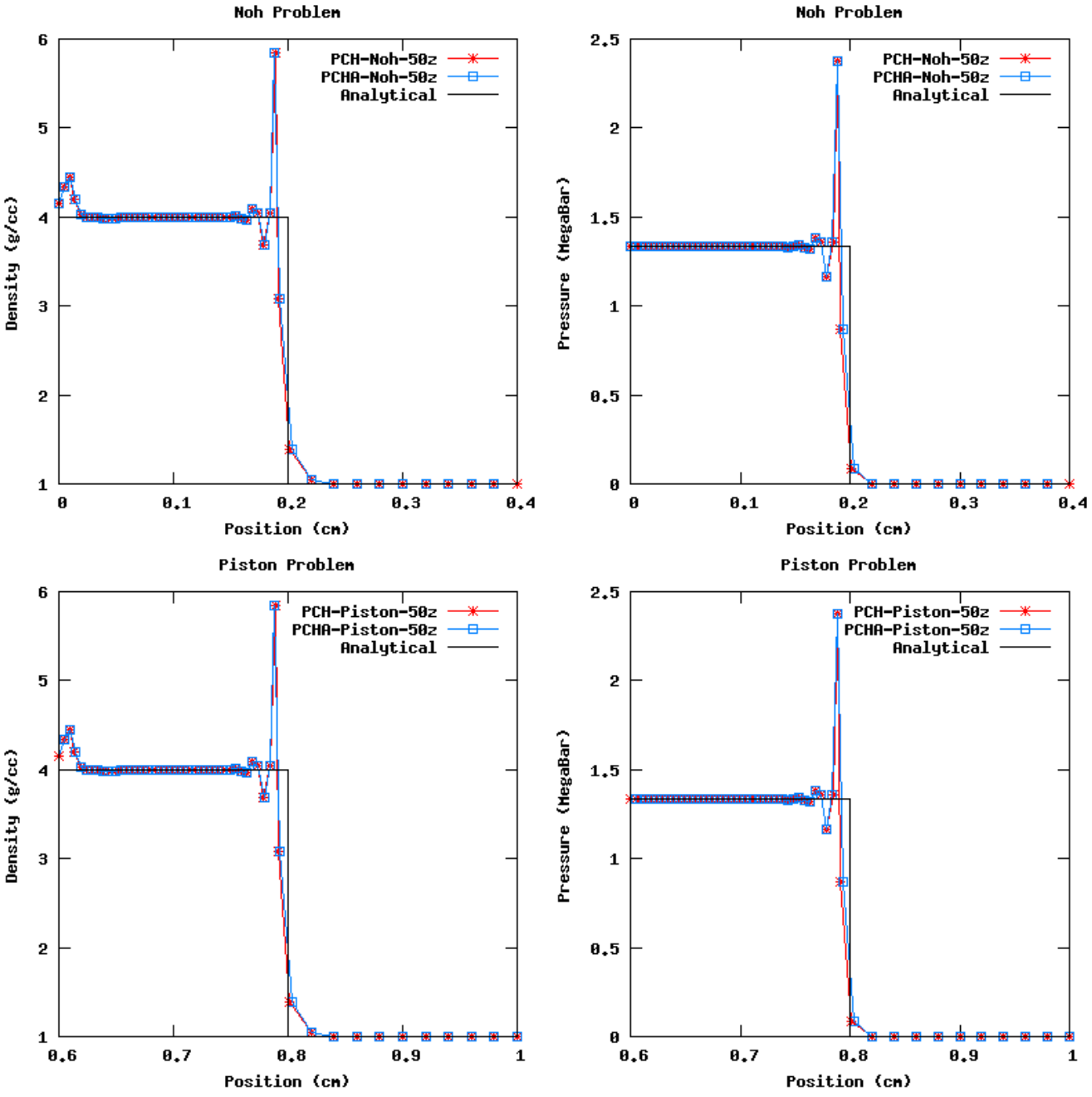


Figure 13. Comparison of the PCH and PCHA methods on the Noh and Piston test problems.

The two methods, PCH and PCHA, yield essentially identical results in the case of the the Noh and Piston test problems.

MPC Comparison to CCH

The MPC method uses an approach very similar to CCH. The results from these two different methods are compared in Figure 14. Both the Sod and the Piston problems were used to compare the two methods.

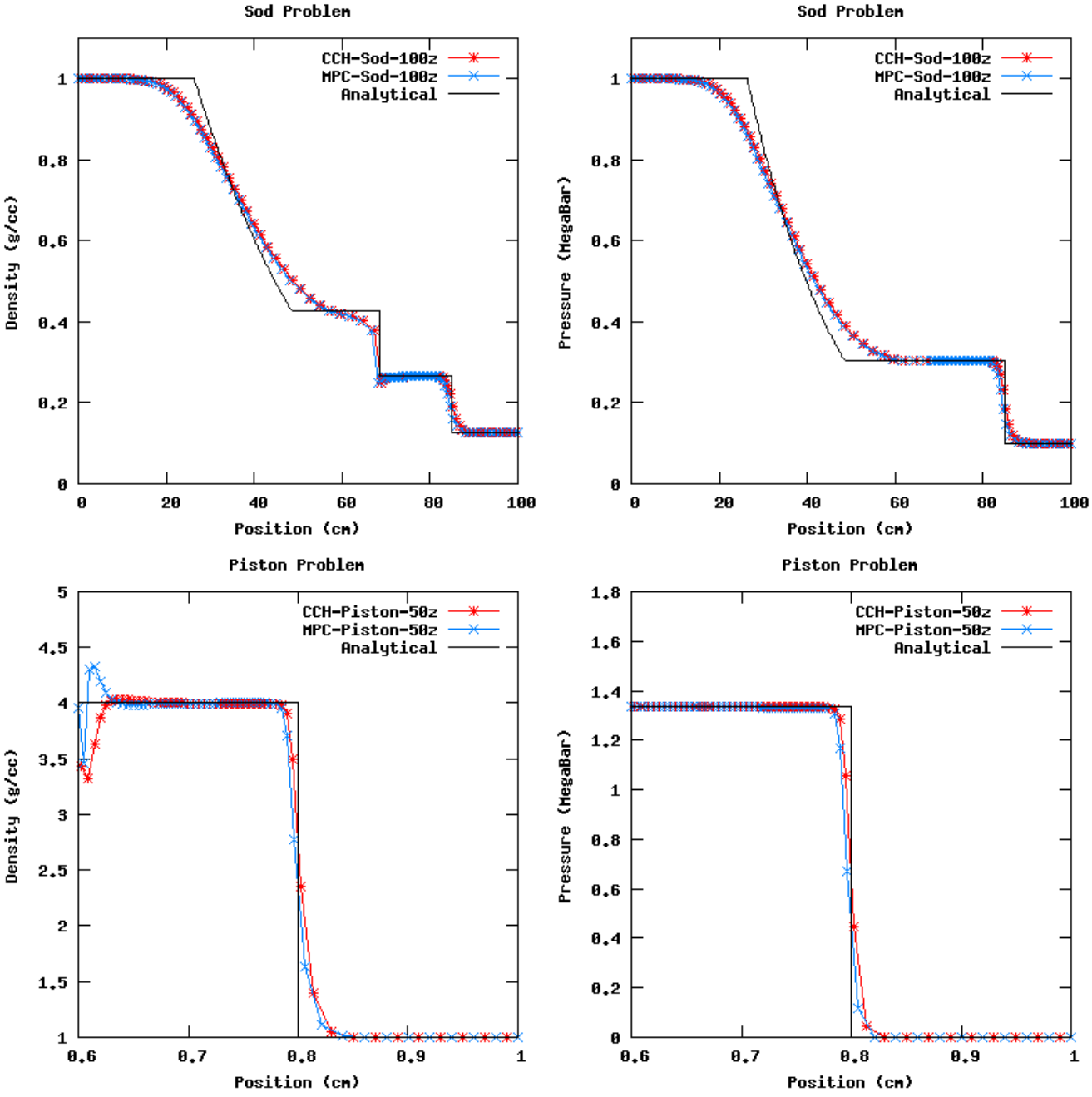


Figure 14. Comparison of the CCH and MPC methods.

The MPC and CCH methods give similar results. The most notable difference is at the fixed boundary for the Piston problem. The differing boundary conditions in MPC and CCH cause considerable differences in the density at this boundary.

Convergence Analysis

The code results from the four methods for the Sod, Piston, and Noh test problems were analyzed at various mesh resolutions to ensure that the calculated results converged to the analytical solutions. Since only first-order methods were used in the 1D hydrocode, near first order convergence is to be expected. For the Piston and Noh test problems, the analytical solutions for pressure, density, and internal energy are piecewise constant functions. For the Sod test problem, portions of the analytical solution are not piecewise constant or linear. For comparison to the hydrocode results, these portions of the analytical solution were resolved using a fixed mesh analytic code [6]. The analytical code produces data points that represent the analytical solution. The results of the hydrocode at various resolutions were mapped to the fixed mesh from the analytical code for comparison. The procedure for this mapping algorithm is as follows.

Since the exact code uses a fixed mesh, the position of any point is given by

$$x_{ex} = i * h$$

where x_{ex} is x-position for the analytical data points, i is the index (0, 1, 2, 3,...) and h is the constant mesh spacing. The appropriate position for a point mapped from the hydrocode mesh, is then

$$i = \frac{x_{calc}}{h}$$

Since the C language always rounds down in float to integer conversion (i.e. 2.24 and 2.99 both round to 2), this equation always ensures that i is the index of the fixed mesh point just below the calculated value, and $i+1$ is the index of the analytical data point just above the calculated value. The expected analytical value and the error are then determined from linear interpolation of the analytical data points. A sample of this algorithm for pressure is as follows.

$$m = \frac{P_{exact(i+1)} - P_{exact(i)}}{x_{exact(i+1)} - x_{exact(i)}}$$

$$P_{interp} = P_{exact(i)} + \Delta x * m$$

where

$$\Delta x = x_{calc} - x_{exact(i)}$$

$$\Delta P = \left| P_{interp} - P_{calc} \right|$$

Figure 15 illustrates this procedure. This same method can be used for analytical solutions without a functional representation, such as the horizontal segments with discontinuities, but this results in a slope of infinite magnitude, making linear interpolation unnecessary.

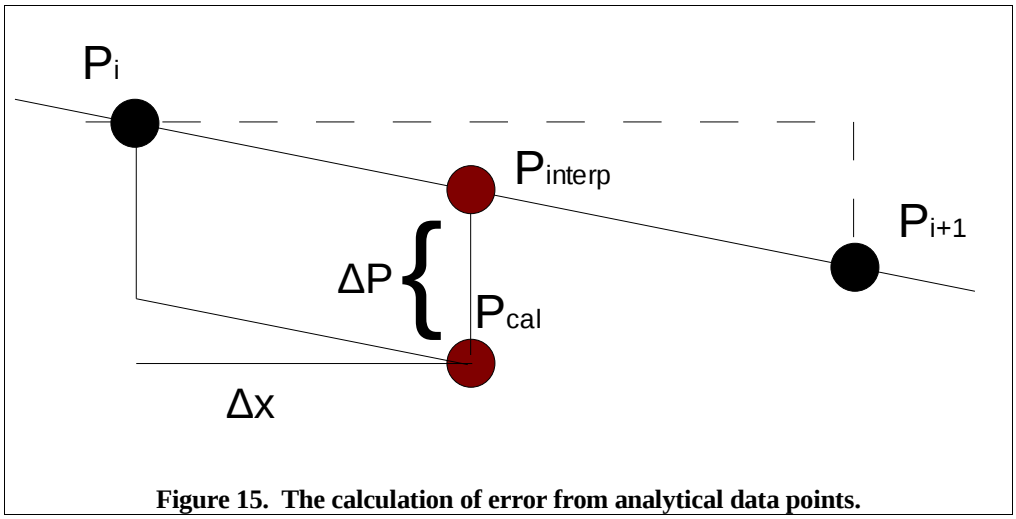


Figure 15. The calculation of error from analytical data points.

Convergence is measured by comparing the volume-weighted combination of all ΔP over the domain with the mesh resolution. Two combinations are common, the L^1 and L^2 norms. The volume weighted L^1 norm is given as:

$$\|L\|^1 = \frac{\sum_i V_i * \Delta P_i}{\sum_i V_i}$$

where V_i is the volume of the Lagrangian element at index i. The volume weighted L^2 norm is

$$\|L\|^2 = \sqrt{\frac{\sum_i (V_i * \Delta P_i)^2}{(\sum_i V_i)^2}}$$

To measure convergence, the rate of decrease in the error with respect to the increase in mesh density is measured. The error is quantified using the L^1 and L^2 norms. The rate decrease of the error should be comparable to the order of the approximations used in the hydrocode. That is, a second order scheme should show a quadratic decrease in error with increasing mesh density. The first order scheme that has been implemented in the 1D hydrocode should show a first order decrease in error with increasing mesh density. Each method is expected to converge according to the following general equation.

$$\epsilon = A\, n^k$$

where ε is the error, A is the convergence coefficient, n is the number of cells, and k is the convergence rate. A power law fit was used to determine the values of A and k that correspond to each method for each test problem given a variety of mesh densities. These results are presented with the corresponding test problem results in the following section.

Test Problems

Sod Problem

The SOD problem simulates the interactions of two materials, one in the first half of the domain, and the other in the second half of the domain. The first material has an initial density of 1 g/cc and internal energy of 2.5 megabar cc/g. The second material has an initial density of 0.125 g/cc and internal energy of 2.25 megabar cc/g. The boundary conditions on both sides of the domain are fixed and the domain is 100 cm in length. The gamma of both materials is set to 1.4. Both materials have an initial velocity of zero and the imaginary barrier between them is removed at t=0. 100 zones were used across the domain to simulate the problem. The results at 20 μs for the four different hydro methods are compared to the analytical solution in Figure 16. SGH and CCH are shown on the left, and MPC and PCHA are shown on the right. The SGH results are slightly more accurate compared the CCH results. The SGH method follows the density discontinuity more accurately than the CCH method at approximately 70 cm. The PCHA method shows numerical ringing about the density discontinuity, while MPC tracks the analytical solution without numerical ringing.

The analytical solution is known [5] and a computer code was used to calculate analytical data points [6]. Convergence studies were carried out for each method. Table 3 shows the results of these convergence studies.

Table 3. Convergence data for the Sod test problem.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^1 Convergence Coefficient
CCH	Pressure	-0.756	0.7821	-1.1207	0.8325
	Density	-0.715	0.5559	-1.0763	0.5727
	Specific Internal Energy	-0.7652	1.8149	1.3086	-1.0401
MPC	Pressure	-0.7589	0.7968	-1.1206	0.8285
	Density	-0.7223	0.5937	-1.0704	0.5581
	Specific Internal Energy	-0.7829	2.1553	-1.0441	1.5312
PCHA	Pressure	-0.844	1.1623	-1.1174	0.7582
	Density	-0.8112	0.8217	-1.0903	0.5526
	Specific Internal Energy	-0.8755	3.17	-1.0642	1.7875
SGH	Pressure	-0.7926	0.6336	-1.1439	0.6956
	Density	-0.7672	0.5212	-1.1025	0.5006
	Specific Internal Energy	-0.8122	1.6637	-1.0348	1.0958

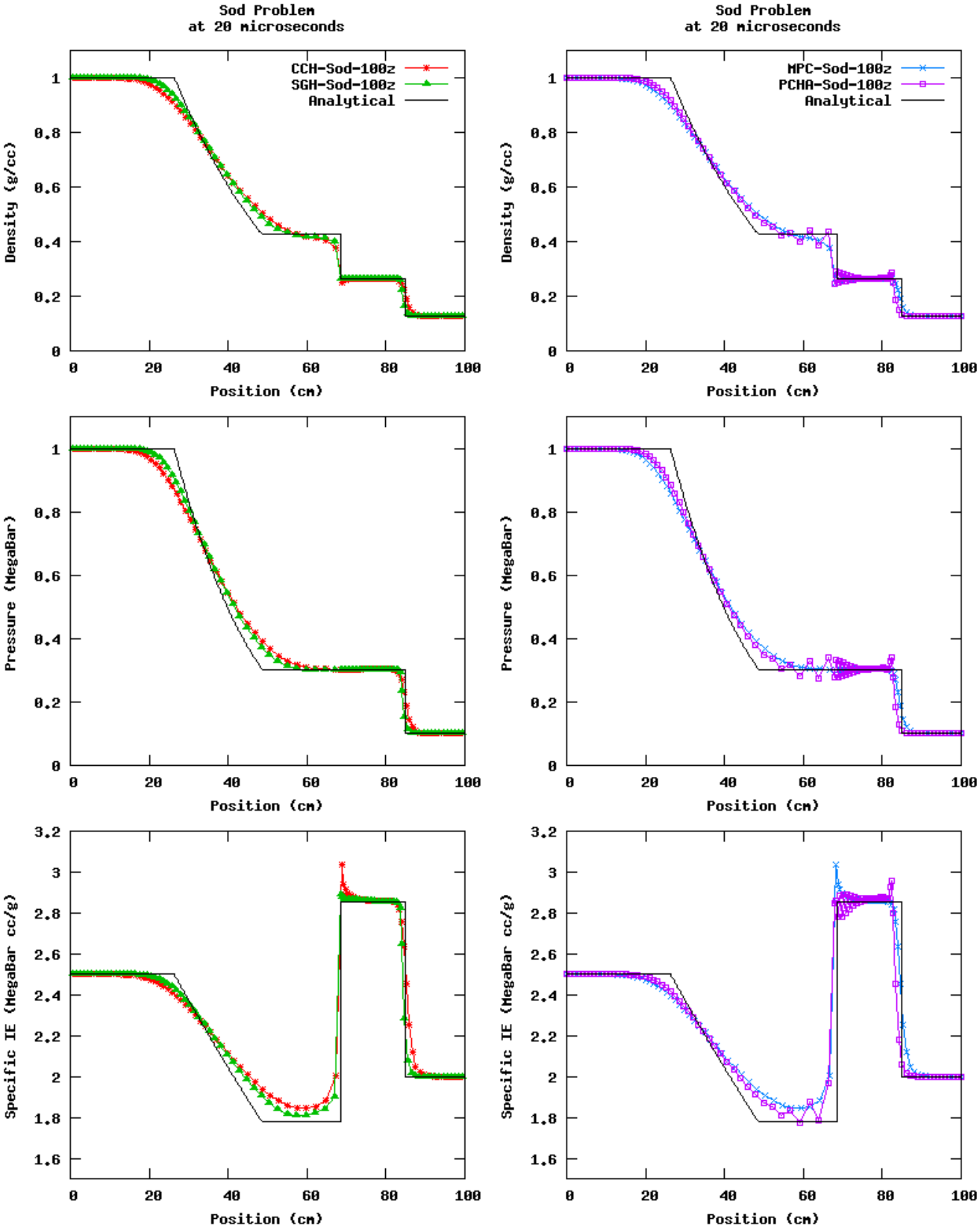


Figure 16. Comparison of the results from four methods for the Sod test problem.

Piston Problem

The piston problem consists of one material across the domain. The initial velocity, initial pressure, and initial internal energy of the material is zero. The initial density of the material is 1 g/cc. The boundary condition on the right is fixed with zero velocity. The boundary condition on the left is fixed with a velocity of 1 cm/μs. The domain is 1 cm in length and 50 zones were used to simulate the problem. The gamma of the material was set to 5/3. The results after 0.6 μs for the four hydro methods are shown in Figure 17. SGH and CCH are shown on the left, and MPC and PCH are shown in the right. The analytical solution to this problem is known. Table 4 shows the results of the convergence studies for this test problem.

Table 4. Convergence data for the Piston test problem.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0145	0.4056	-1.0065	0.2534
	Density	-1.0015	1.8994	-1.0065	0.8630
	Specific Internal Energy	-1.0011	0.2905	-1.0085	0.1621
MPC	Pressure	-1.0120	0.4494	-1.0051	0.2966
	Density	-0.9995	1.6330	-1.0058	0.7468
	Specific Internal Energy	-0.9978	0.2024	-1.0054	0.1040
PCH	Pressure	-1.0101	0.5734	-1.0109	0.3007
	Density	-1.0021	1.5545	-1.0101	0.6628
	Specific Internal Energy	-0.9935	0.1702	-0.9999	0.0876
SGH	Pressure	-1.0018	0.3192	-1.0003	0.2827
	Density	-0.9962	0.9938	-1.0026	0.5665
	Specific Internal Energy	-0.9978	0.1068	-0.9934	0.0655

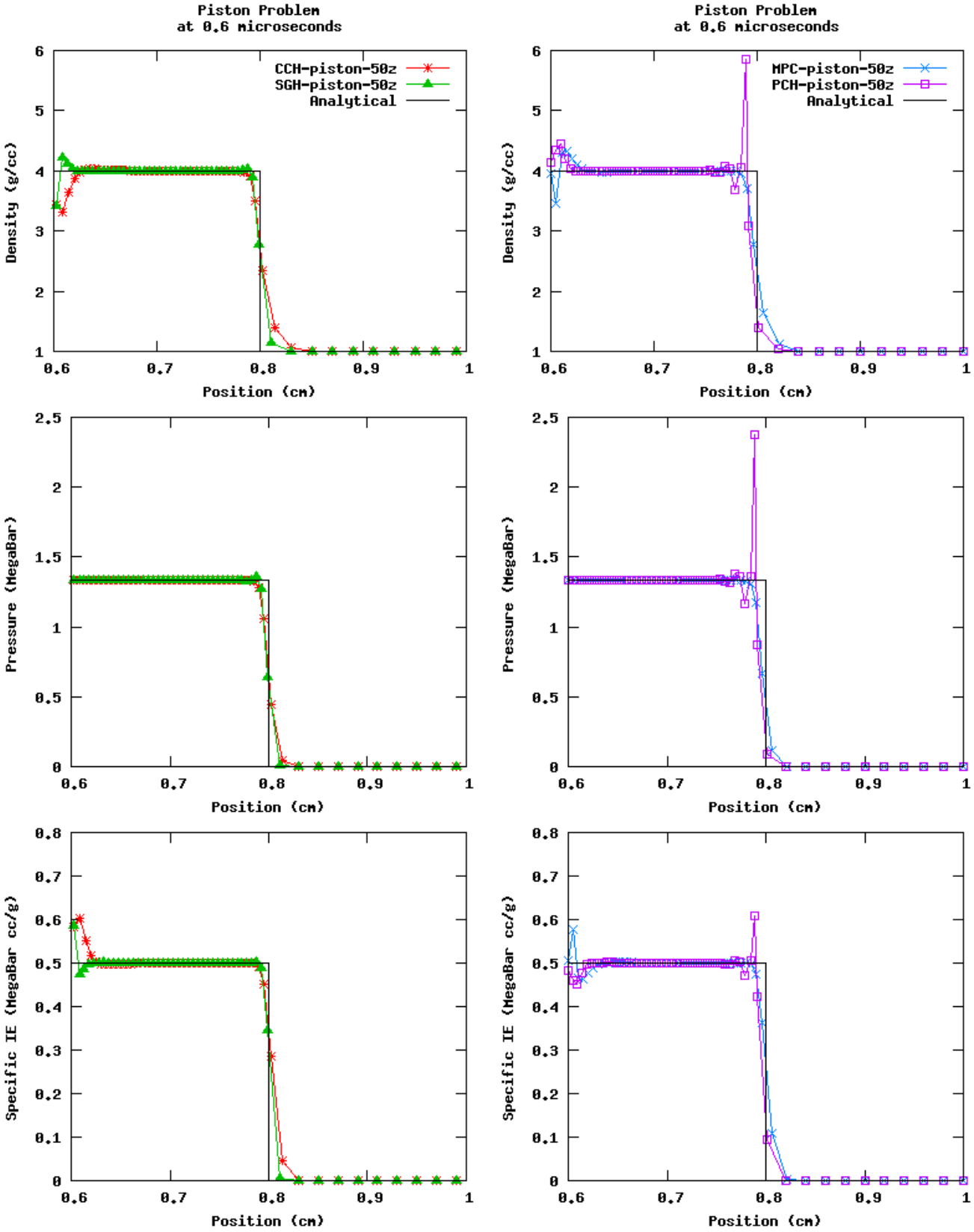


Figure 17. Comparison of results from four methods for the Piston test problem.

Noh

The Noh problem involves a single material moving at -1 cm/μs and colliding with a fixed boundary at the origin. The domain for this simulation was set to 1 cm and 50 zones were used to simulate the problem. The boundary at the origin is fixed and the boundary at the end is moving at -1 cm/μs with the fluid. The results are for a 1D simulation in planar, cylindrical, and spherical coordinates with gamma equal to 5/3. The analytical solution is known [7].

Planar Coordinates

The results for planar coordinates are shown in Figure 18. Convergence data is recorded in Table 5.

Table 5. Convergence data for the Noh test problem in planar coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0145	0.4056	-1.0065	0.2534
	Density	-1.0015	1.8994	-1.0065	0.8630
	Specific Internal Energy	-1.0011	0.2905	-1.0085	0.1621
MPC	Pressure	-1.0096	0.4392	-1.0065	0.2947
	Density	-0.9983	1.6114	-1.0054	0.7394
	Specific Internal Energy	-0.9980	0.2027	-1.0061	0.1047
PCH	Pressure	-1.0028	0.5569	-1.0049	0.2941
	Density	-0.9975	1.5221	-1.0040	0.6450
	Specific Internal Energy	-0.9938	0.1707	-0.9988	0.0863
SGH	Pressure	-1.0018	0.3192	-1.0003	0.2827
	Density	-0.9962	0.9938	-1.0026	0.5665
	Specific Internal Energy	-0.9954	0.1128	-1.0003	0.0699

Cylindrical Coordinates

The results for cylindrical coordinates are shown in Figure 19. The results of the convergence studies for the Noh problem in cylindrical coordinates are shown in Table 6.

Table 6. Convergence data for the Noh test problem in cylindrical coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0068	5.7774	-1.0098	2.1002
	Specific Internal Energy	-0.8970	0.6561	-1.0037	0.3299
MPC	Pressure	-1.0169	7.0382	-0.9940	2.5974
	Specific Internal Energy	-0.88	0.4887	-1.0084	0.2416
PCHA	Pressure	-0.9901	5.8003	-0.9903	3.1239
	Specific Internal Energy	-0.8482	0.5744	-0.9986	0.2265
SGH	Pressure	-0.9888	16.9188	-0.9235	6.4584
	Specific Internal Energy	-0.7903	1.4122	-0.9314	0.7030

Spherical Coordinates

The results for spherical coordinates are shown in Figure 20. The results of the convergence studies are shown in Table 7.

Table 7. Convergence data for the Noh test problem in spherical coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-0.9721	31.4138	-1.0191	9.3409
	Specific Internal Energy	-0.8611	0.7734	-1.0023	0.3596
MPC	Pressure	0.0477	5.3956	-0.4547	6.1347
	Specific Internal Energy	-0.8986	2.8903	-0.9886	1.4213
PCHA	Pressure	0.0516	5.2193	-0.4507	5.9481
	Specific Internal Energy	-0.8696	2.3708	-0.9904	1.0447
SGH	Pressure	-1.0074	99.1013	-0.9668	32.2163
	Specific Internal Energy	-0.7842	2.3395	-0.9812	1.3969

It is interesting to note that for spherical coordinates, the convergence rate for the point centered methods (MPC and PCHA) are positive for pressure in the L^1 norm. This is a reason to further investigate curvilinear coordinates for all the hydro methods presented. The results for a mesh resolution of 10,000 zones is shown in Figure 21. It is clear that the MPC and PCHA methods converge to a value for pressure near the wall that is less than the analytical value.

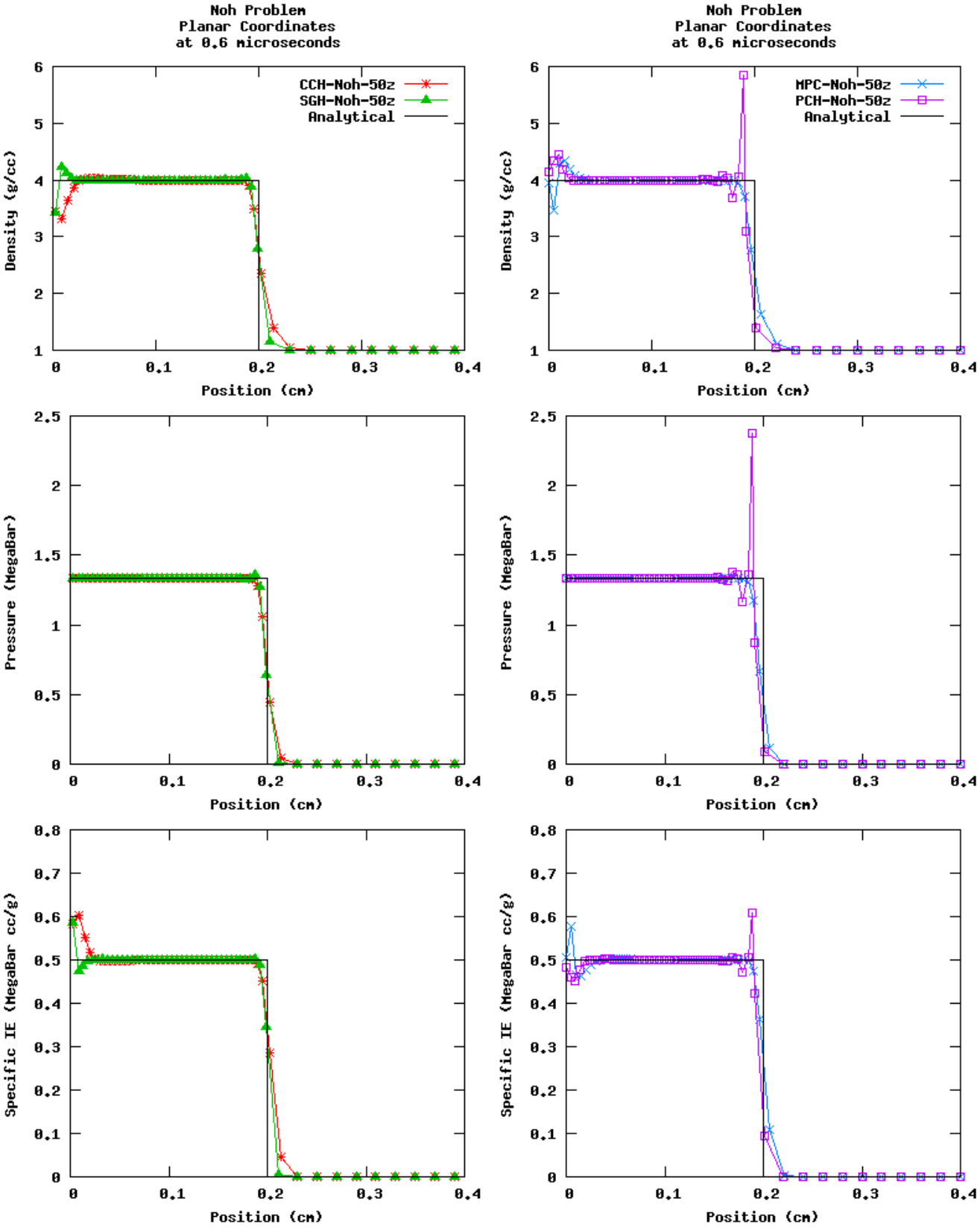


Figure 18. Comparison of results from four methods for the Noh test problem in planar coordinates.

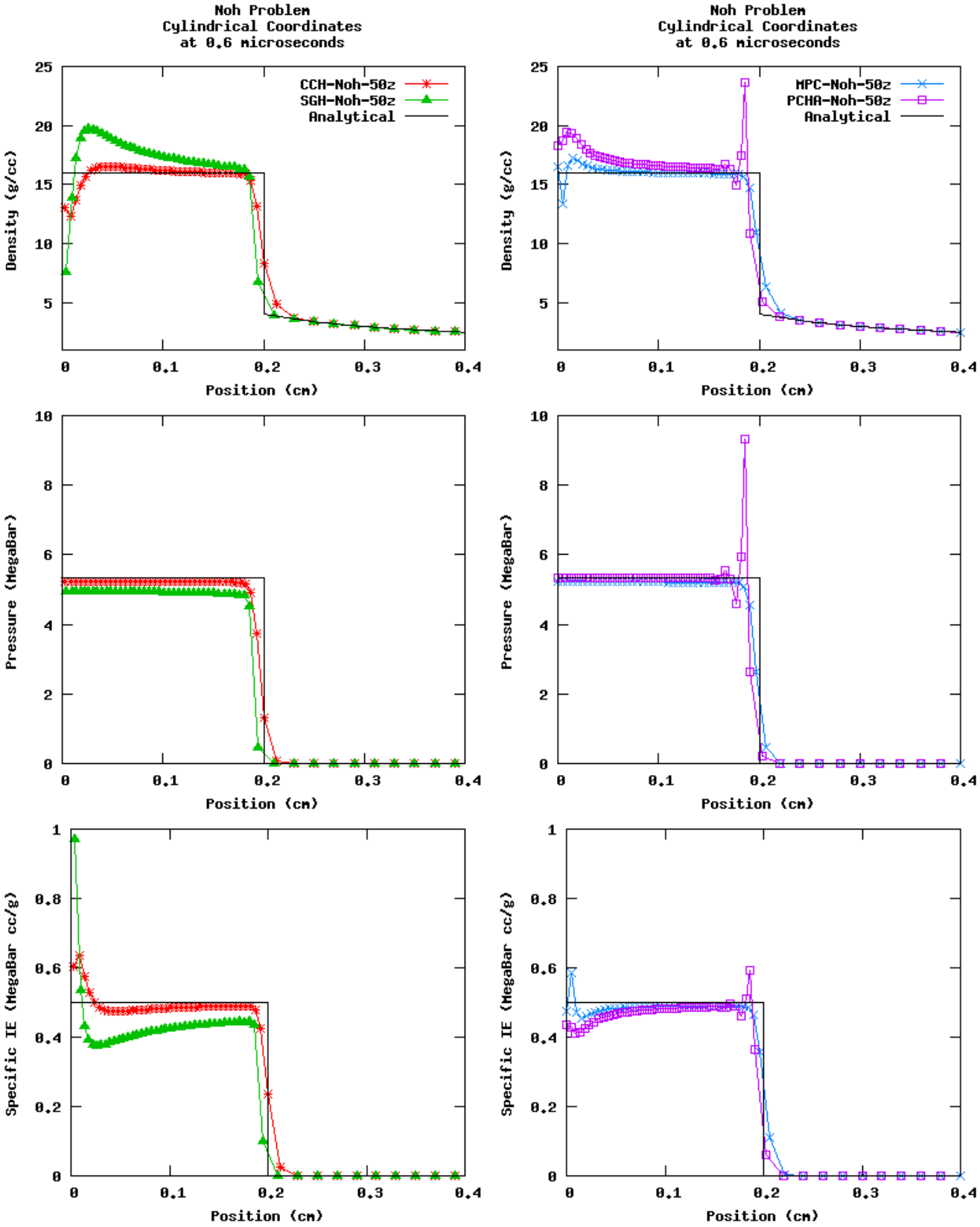


Figure 19. Comparison of results from four methods for the Noh test problem in cylindrical coordinates.

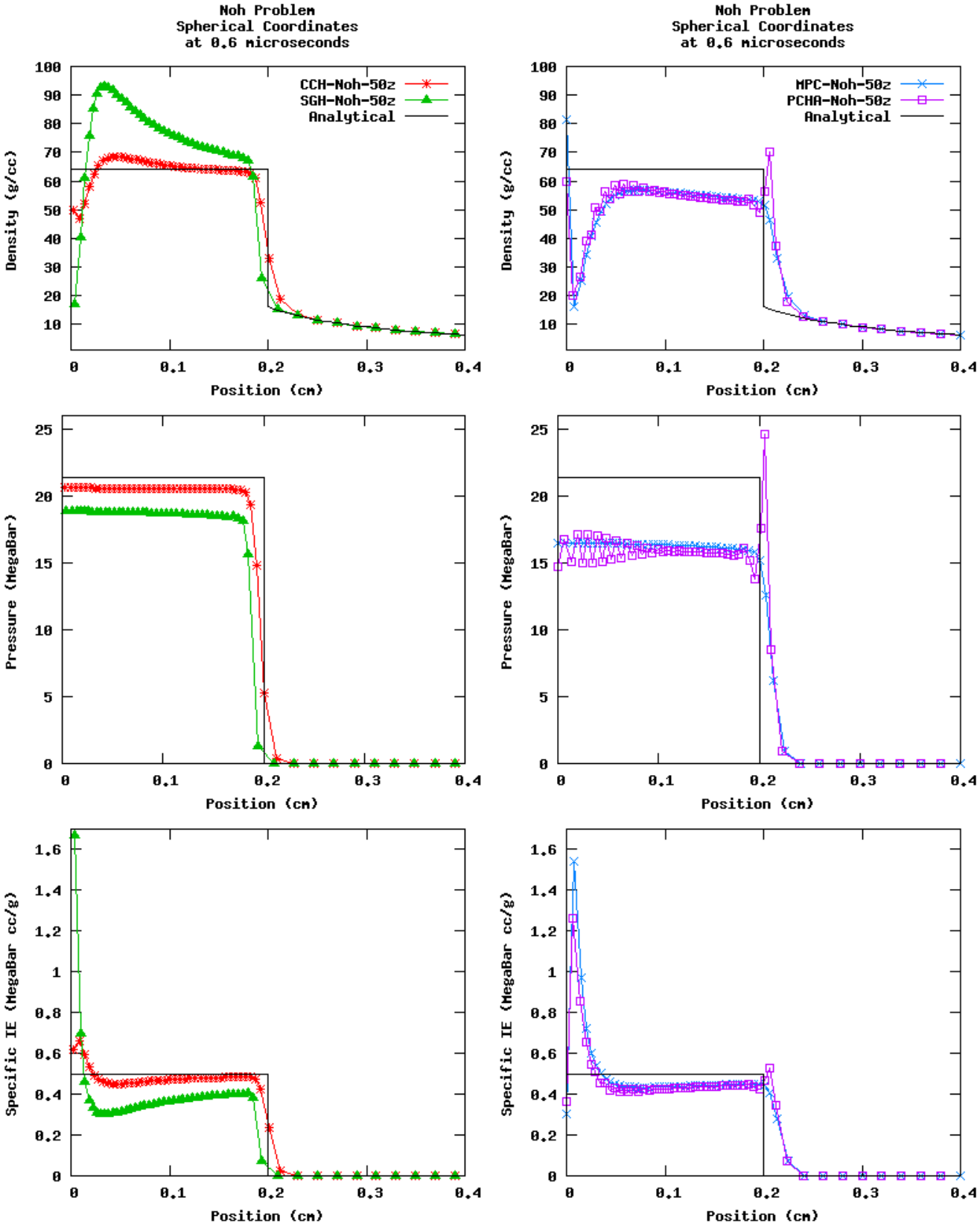


Figure 20. Comparison of results from four methods for the Noh test problem in spherical coordinates.

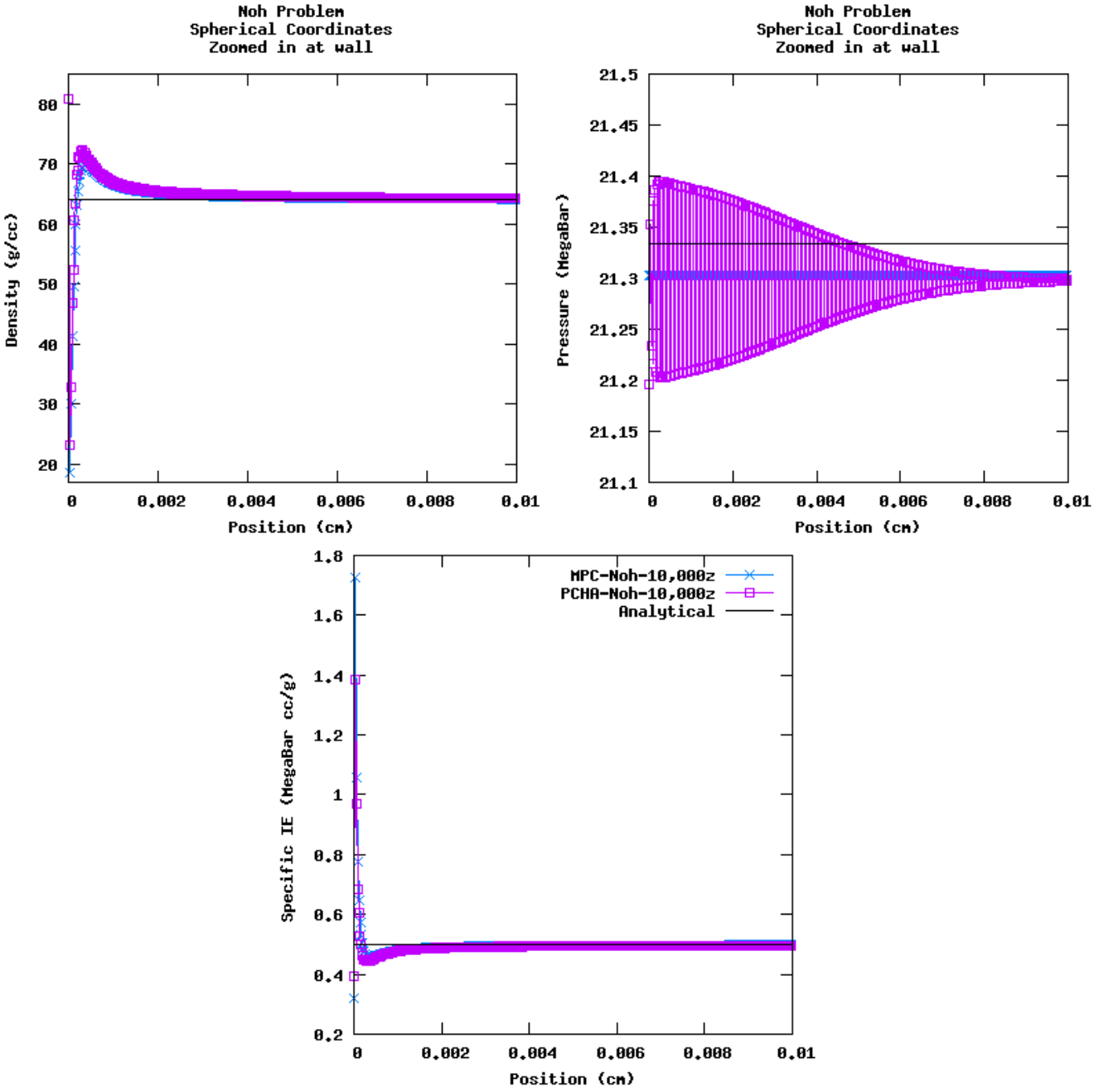


Figure 21. MPC and PCHA results for the Sedov test problem, zoomed at the wall.

The PCHA method produces numerical oscillations at the wall. Both the MPC and PCHA methods converge to a lower pressure near the wall than the analytical solution demands.

Sedov

The Sedov problem involves a blast propagating from the origin. The simulation is initialized with a specified amount of internal energy deposited in the cell at the origin. Once the simulation is started, a shock wave moves away from the origin. The domain for this simulation was set to 1.2 cm and 60 zones were used to simulate the problem. The gamma of the gas was set to 5/3 across the domain. The density is initially set to 1 g/cc across the domain, and the pressure and internal energy are set initially to zero everywhere except at the origin. The analytical solution is known [8] and the analytical data points for each simulation were computed using an analytical code [9]. The simulation was run in planar coordinates. In cylindrical and spherical coordinates, the simulation produced inconsistent results. Therefore, these results are not shown. Further investigation into the methods used in the 1D hydrocode for cylindrical and spherical coordinates is recommended.

Planar Coordinates

For planar coordinates, the amount of extensive internal energy deposited in the cell at the origin was 0.3 megabar cc. With this amount of energy, the exact form of the shock is located at approximately 1 cm after 1μs. The extensive internal energy was divided by the mass of the cell at the origin and applied to the cell at the origin as specific internal energy. The volume of the cell at the origin is initially 0.02 cc, therefore, the amount of specific internal energy deposited in the cell at the origin is 15 megabar cc/g. The results for planar coordinates are shown in Figure 22. SGH and CCH are shown on the left, and MPC and PCHA are shown together on the right. The PCH method experiences cell collapse in the Sedov test problem and the simulation stalls. Therefore no results were obtained from the PCH method for the Sedov test problem.

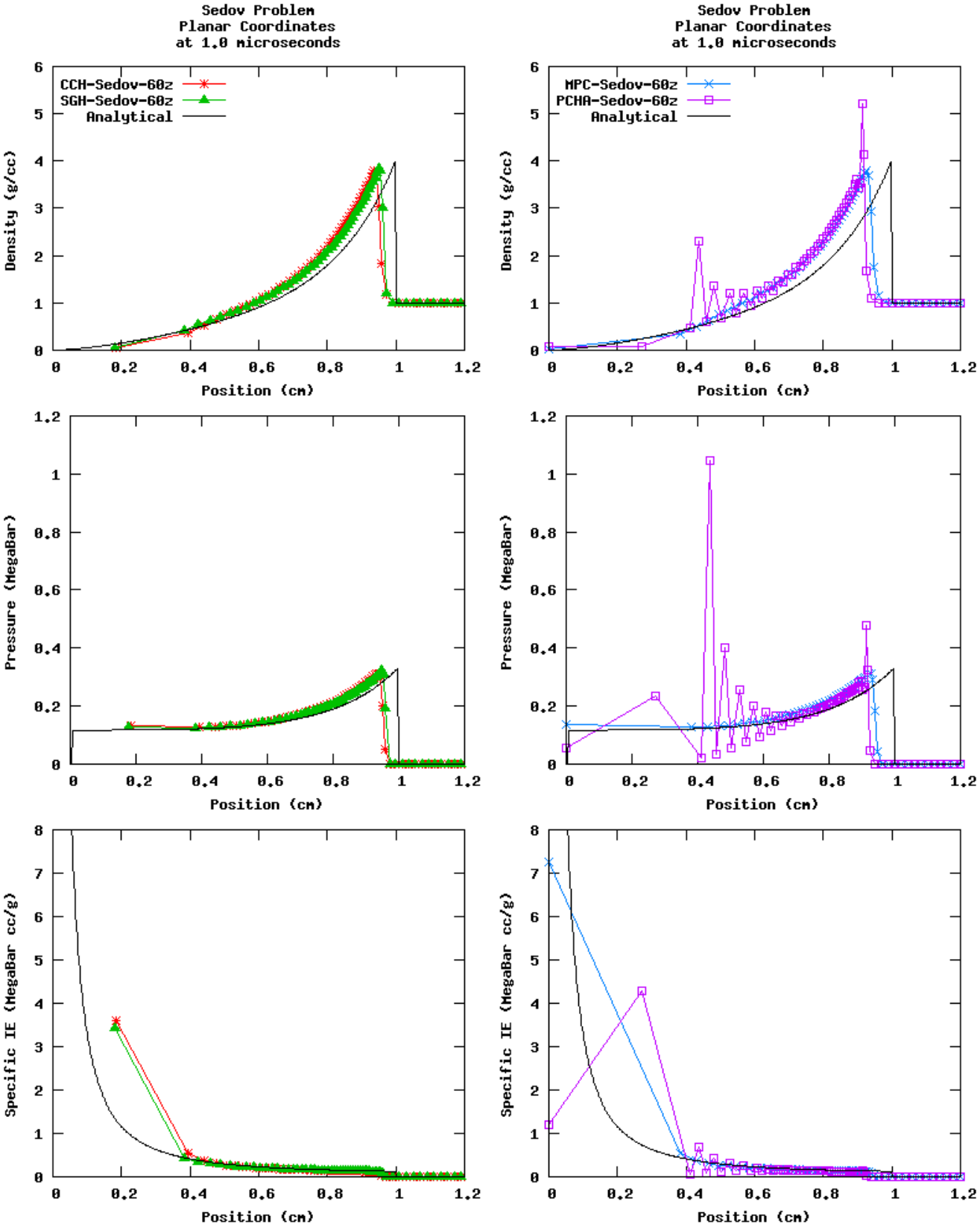


Figure 22. Comparison of results from four methods for the Sedov test problem in planar coordinates.

The results for the PCHA method for the Sedov test problem clearly demonstrate the numerical ringing associated with this method. Qualitative convergence studies were performed for the Sedov problem in planar coordinates. The results are shown in Figure 23. The CCH and SGH methods converge to the analytical solution as expected. The MPC method also converges. However, the numerical ringing in the PCHA method produces errors of greater magnitude in pressure with a mesh of 300 zones compared to the mesh of 60 zones in Figure 22. It is also interesting to note that the PCHA method does not converge to the analytical solution as the mesh is refined. The shock is still slow, as predicted with a mesh of 60 zones, even with a mesh of 300 zones.

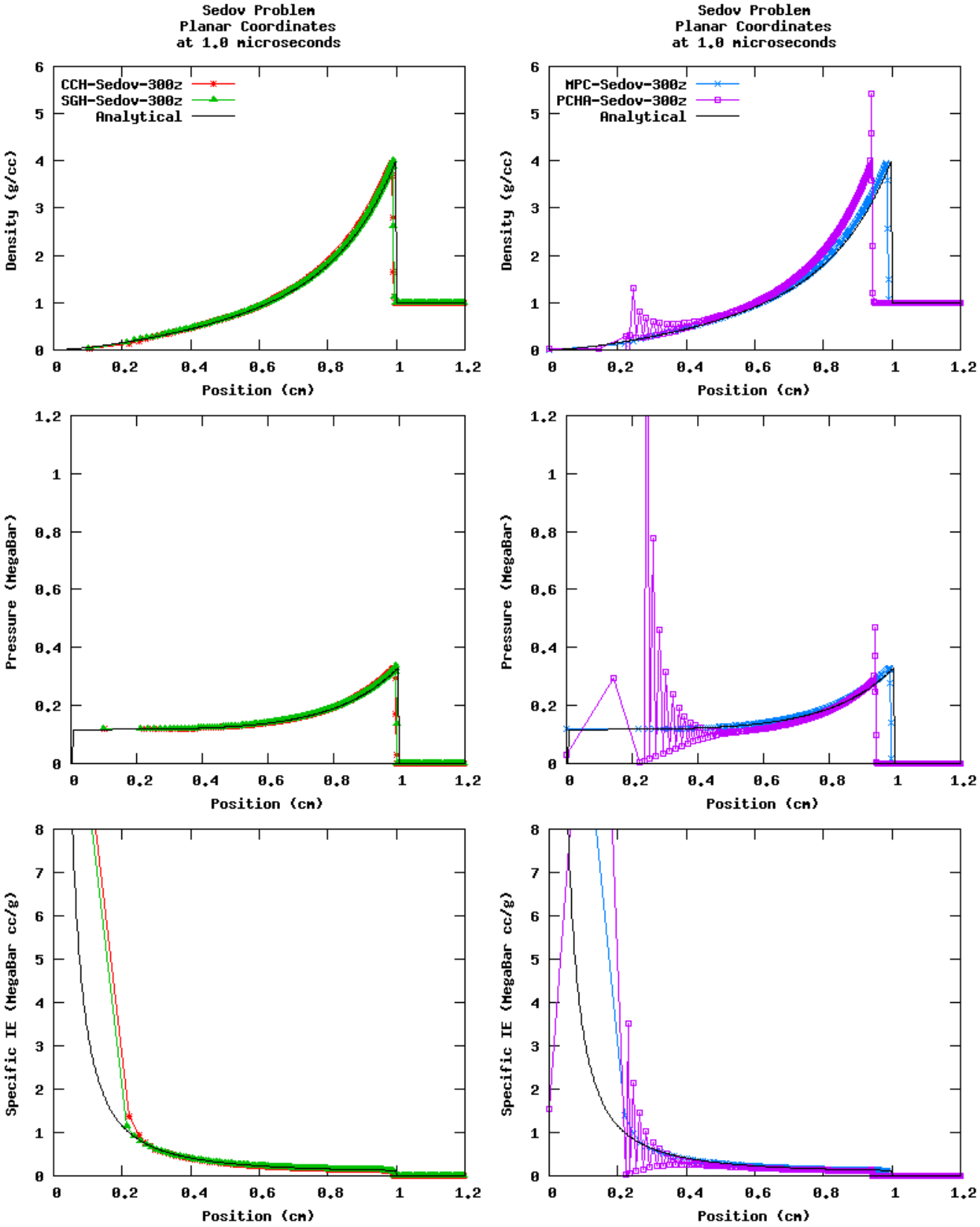


Figure 23. Effect of mesh refinement on the results for the Sedov test problem in planar coordinates.

Conclusions

The project focused on developing a 1D hydrocode to simulate gas dynamics using three different discretization techniques. The techniques used were cell-centered (CCH), staggered grid (SGH), and point-centered hydrodynamics (PCH). The PCH method proved to fail under strong shock conditions. The failure was induced by zone collapse and a diminishing time step. Two additional PCH methods were developed to mitigate this failure. These methods are called modified PCH (MPC) and averaged PCH (PCHA). These two methods produced results for all test problems considered. MPC is similar to CCH, simply shifted by half a cell. PCHA produces numerical oscillations at shock discontinuities.

Four test problems were used to verify the hydrocode. These problems were the Sod, Piston, Noh, and Sedov problems. For the Sod, Piston, and Noh problems, the convergence rates were computed for each method. In planar coordinates, all convergence rates were consistent with the first order numerical scheme used in the code. However, in spherical coordinates for the Noh problem, the pressure did not converge to the analytical solution. Further investigation into curvilinear coordinates is recommended for future work.

The CCH, SGH, and MPC methods showed qualitative convergence in the Sedov test problem in planar coordinates. In curvilinear coordinates, the 1D hydrocode did not produce consistent results for the Sedov test problem. The numerical oscillations produced by the PCHA method are more apparent in the Sedov test problem than in other test problems.

In general, the SGH method performs better for the Sod and Sedov test problems. The CCH method performs better for the Piston and Noh test problems. It is recommended that the different PCH methods be investigated further in order to evaluate the merits of each method and perhaps continue their development.

Future Work

Curvilinear Coordinates

For the Sedov test problem in cylindrical and spherical coordinates, the simulation did not converge to a solution. It was found that as the cell size decreased and the amount of extensive internal energy deposited in the cell at the origin remained constant, the simulation produced varying results. Also, the simulation did not conserve total energy over the course of the simulation. In the future, this issue needs to be studied, and solution needs to be determined. Some hypotheses have been formed. The 1D hydrocode conserves total energy in the Sedov test problem in planar coordinates, but not in cylindrical or spherical coordinates. Therefore, it seems most likely that the error in energy conservation is due to the calculation of the areas and volumes in cylindrical and spherical coordinates. Alternatively, the way the boundary conditions are formulated could be resulting in error for cylindrical and spherical coordinates.

Second Order Scheme

The addition of a second order scheme would involve adjusting the Riemann solver in each of the hydro method programs. Instead of assigning a value of pressure or velocity to an interface from

the nearest control volume, the pressure and velocity are projected via a central gradient. The projection of the velocity is illustrated in Figure 24.

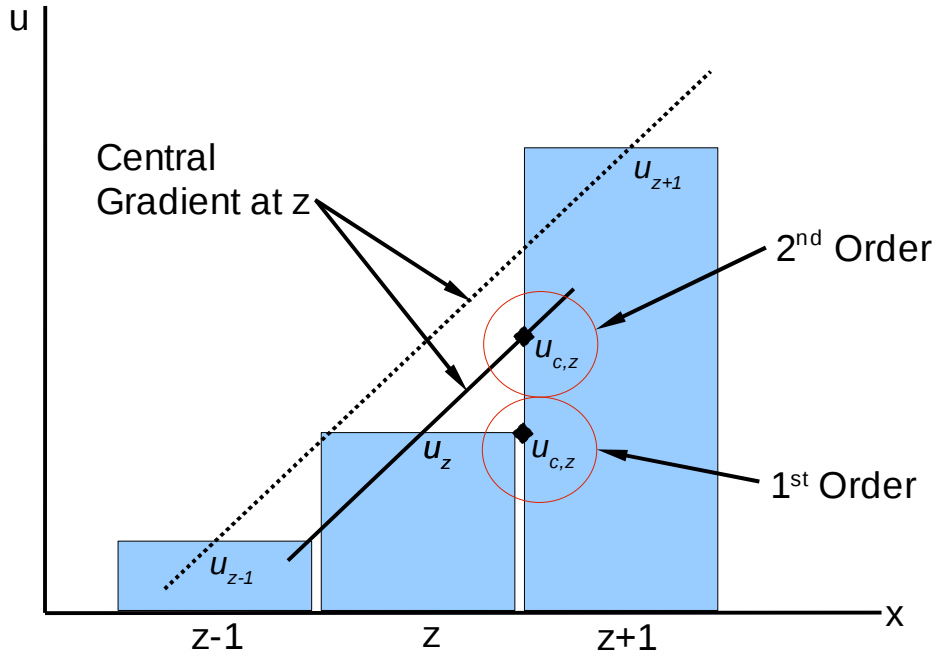


Figure 24. Second order velocity projection.

$u_{c,z}$ is evaluated at the cell interfaces using a gradient in a second order scheme. Assuming a constant value across the control volume is a first order approximation. The central gradient is given by

$$\bigtriangledown u = \frac{u_{z+1} - u_{z-1}}{\frac{1}{2}(\Delta x)_{z-1} + (\Delta x)_z + \frac{1}{2}(\Delta x)_{z+1}}$$

and $u_{c,z}$ is given by

$$u_{c,z} = u_z + \frac{1}{2}(\Delta x)_z(\bigtriangledown u)$$

Limiters will also need to be used in a second order scheme. Limiters are used to limit the gradient if the gradient projects a value that is greater than the highest value or less than the lowest value. Figure 25 illustrates this concept, where $\theta_{z,p}$ is a limiter.

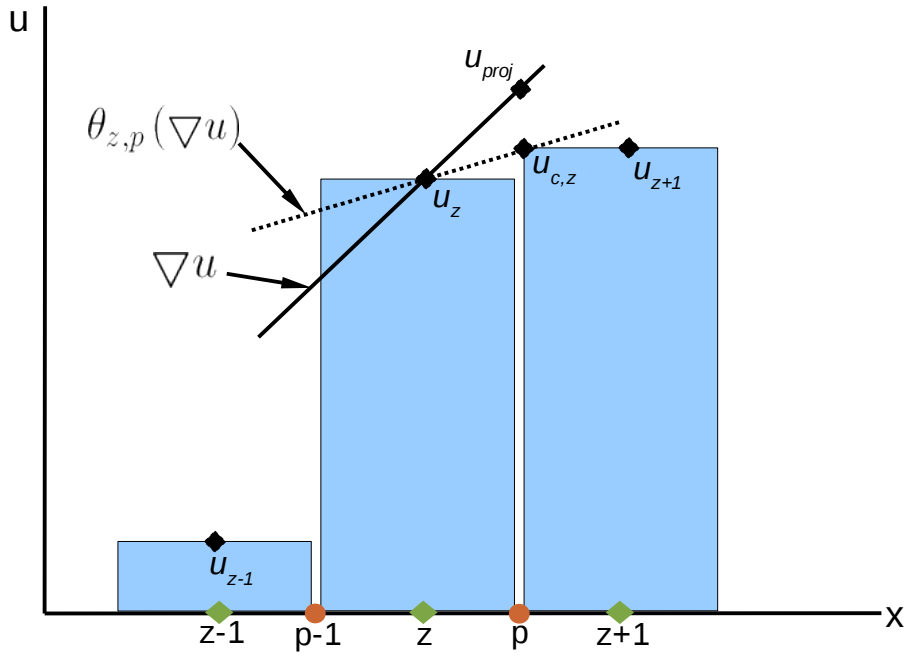


Figure 25. Second order velocity projection with a gradient limiter.

The limiter is computed using the formula

$$\theta_{z,p} = \min\left[\frac{u_z - u_{z+1}}{u_z - u_{proj}}, 1\right]$$

Similarly, at the interface at *p-1*, the limiter would be evaluated by

$$\theta_{z,p-1} = \min\left[\frac{u_z - u_{z-1}}{u_z - u_{proj}}, 1\right]$$

The projected value, or corner value, at *p* can then be computed using

$$u_{c,z} = u_z + \frac{1}{2}(\Delta x)_z \; \theta_{z,p}(\nabla u)$$

The corner values computed with the limited gradients are then used in the Riemann solver.

References

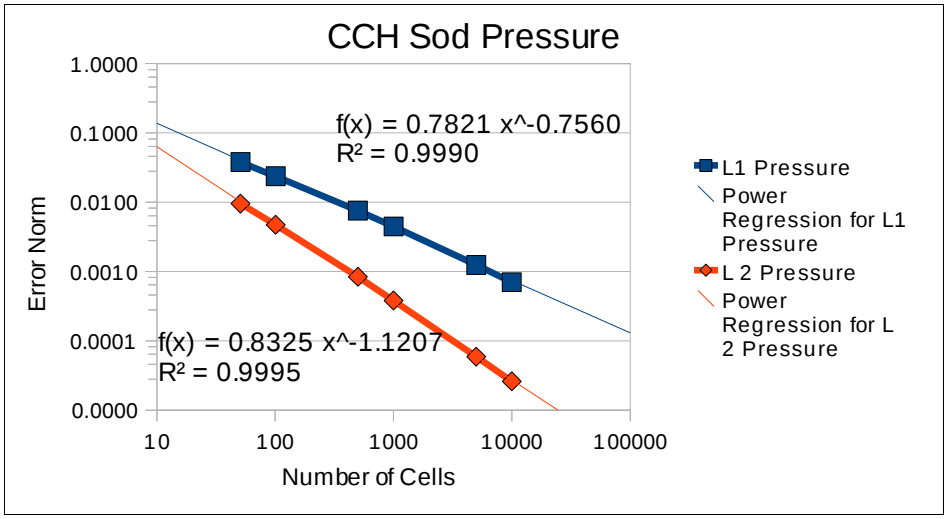
[1] R. Loubere, M. Shashkov, B. Wendroff, Volume consistency in a staggered grid Lagrangian hydrodynamics scheme. Journal of Computational Physics 227 (2008); 3731-3737.
[2] J. Dukowicz, A general, non-iterative Riemann solver for Godunovs method. Journal of Computational Physics 1985; 61:119-137.
[3] N. Morgan, A Lagrangian Staggered Grid Godunov-like Approach. Not yet published.
[4] N. Morgan, A dissipation model for staggered grid Lagrangian hydrodynamics. Computers and Fluids 2013; 83:48-57.
[5] G.A. Sod, Survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. Journal of Computational Physics 26 (1978).
[6] sod.py (Python Code for Sod Solution). Written by Scott Doebling, Los Alamos National Lab, February 2013.
[7] W.F. Noh, Errors for calculations of strong shocks using artificial viscosity and an artificial heat flux. Journal of Computational Physics 72 (1987); 78–120.
[8] L. I. Sedov, Propagation of strong shock waves, Journal of Applied Mathematics and Mechanics 10 (1946), 241–250.
[9] sedov_exact.py (Python Code for Sedov Solution). Written by Scott Doebling, Los Alamos National Lab, August 2012.

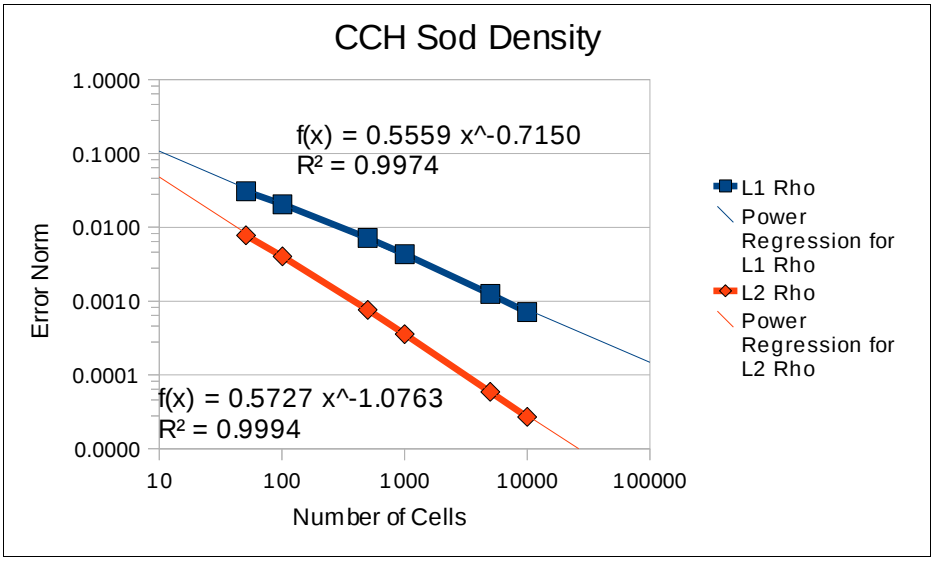
Appendix

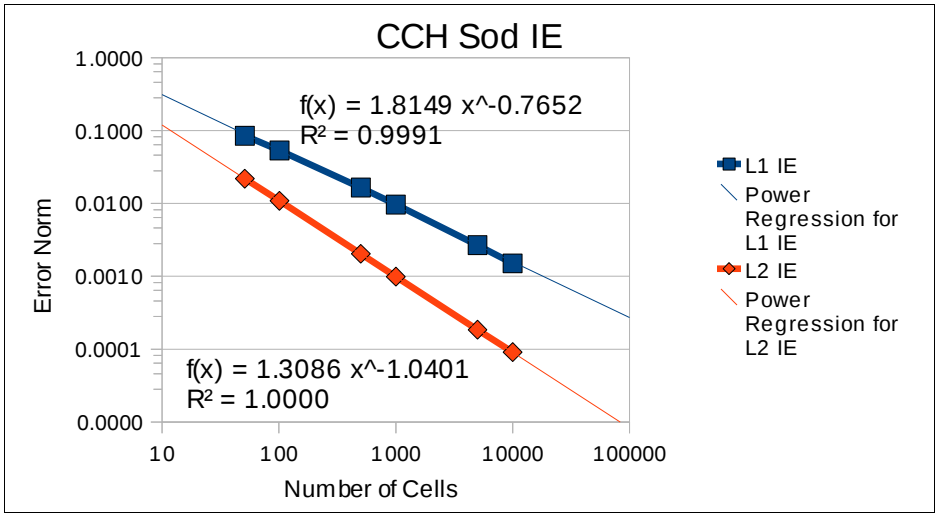
Convergence Plots

SOD

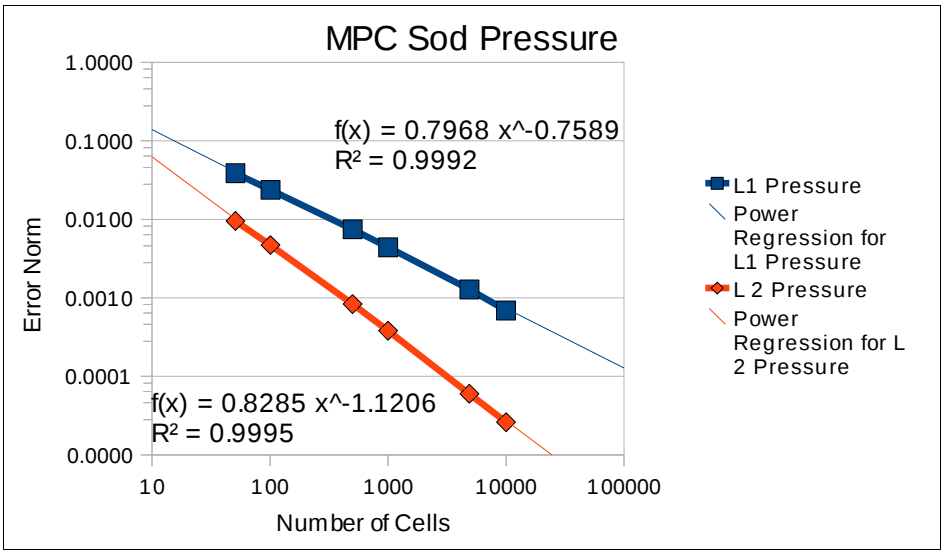
CCH

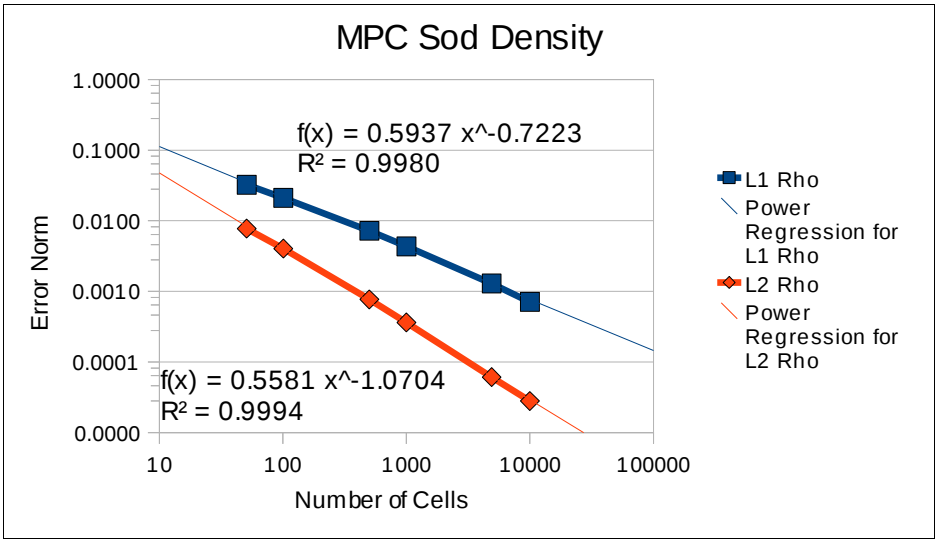


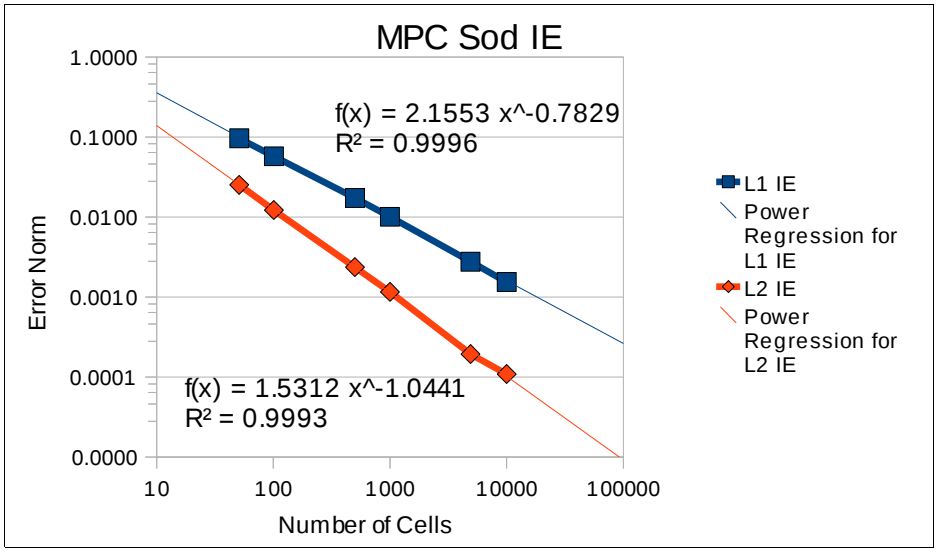



CCH Sod IE
f(x) = 1.8149 x^-0.7652
R² = 0.9991
f(x) = 1.3086 x^-1.0401
R² = 1.0000
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

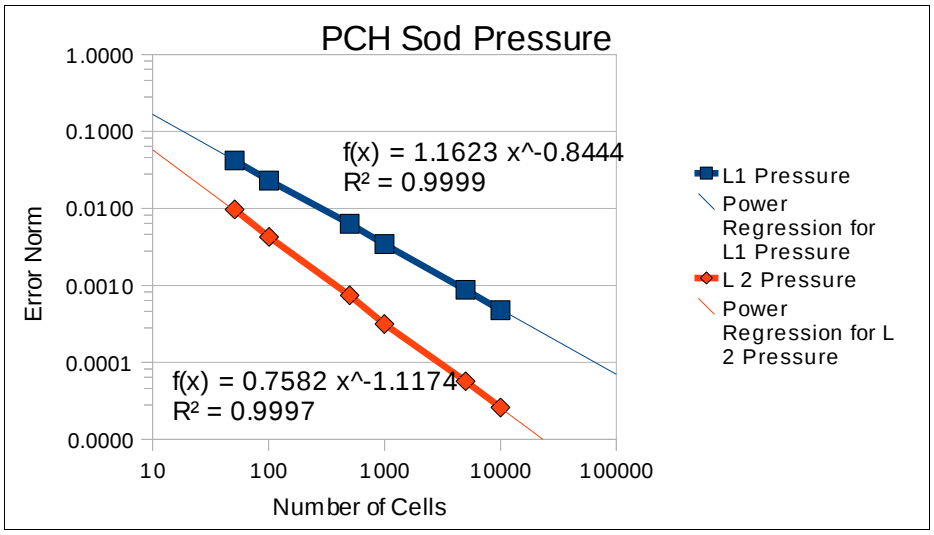
MPC

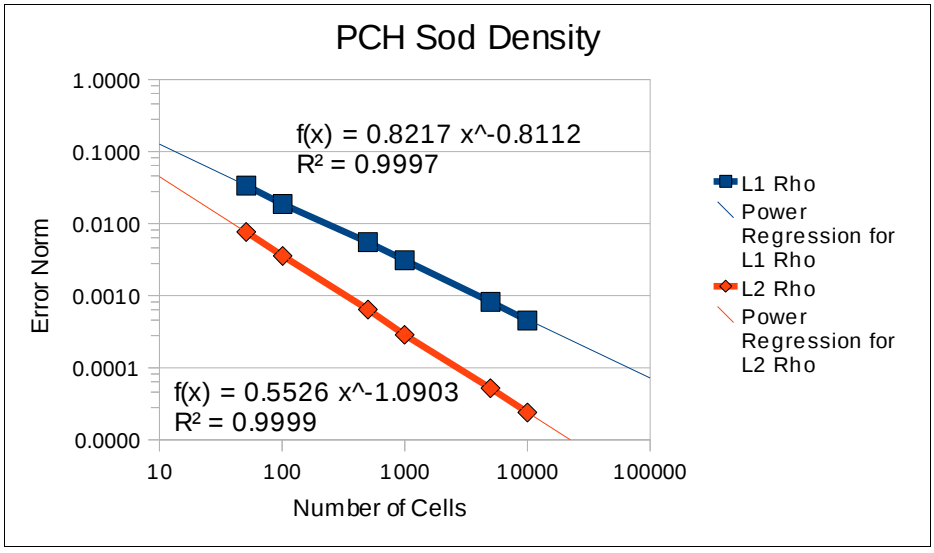

MPC Sod Pressure
f(x) = 0.7968 x^-0.7589
R² = 0.9992
f(x) = 0.8285 x^-1.1206
R² = 0.9995
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

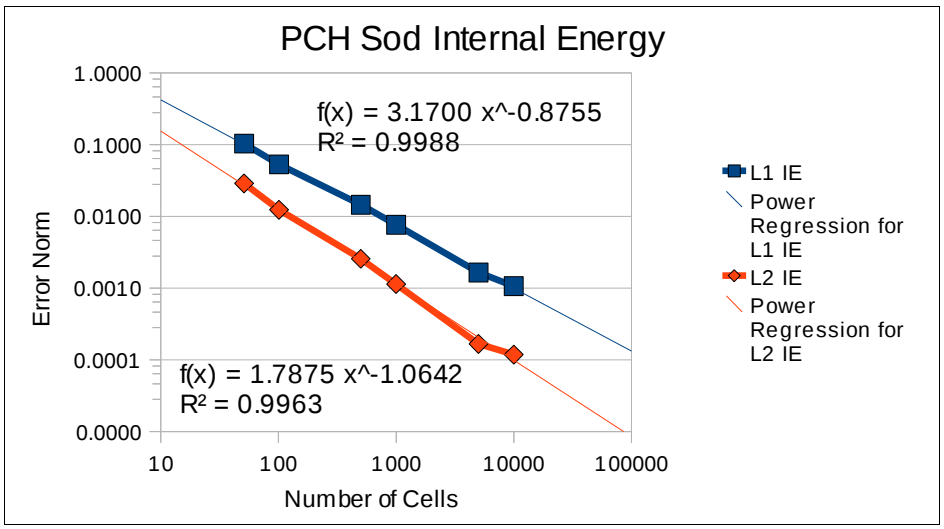

MPC Sod Density
f(x) = 0.5937 x^-0.7223
R² = 0.9980
f(x) = 0.5581 x^-1.0704
R² = 0.9994
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


MPC Sod IE
f(x) = 2.1553 x^-0.7829
R² = 0.9996
f(x) = 1.5312 x^-1.0441
R² = 0.9993
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

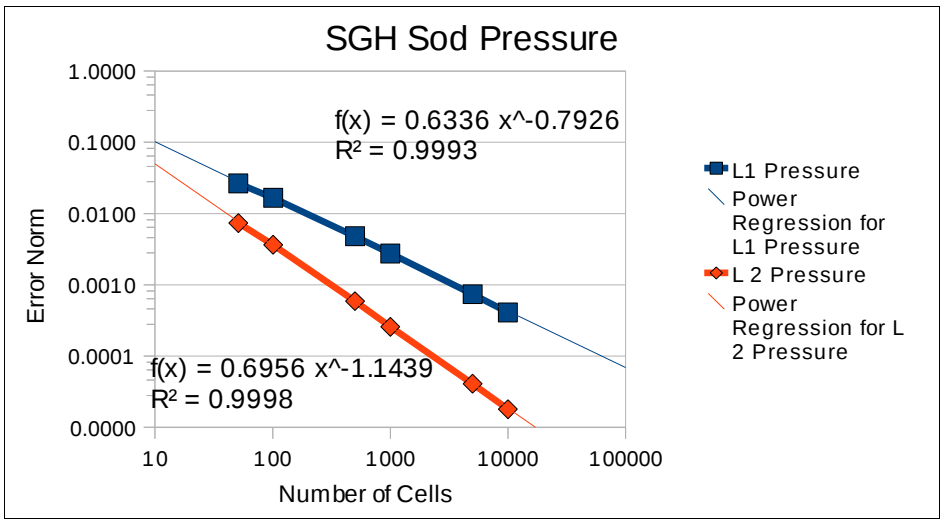
PCHA

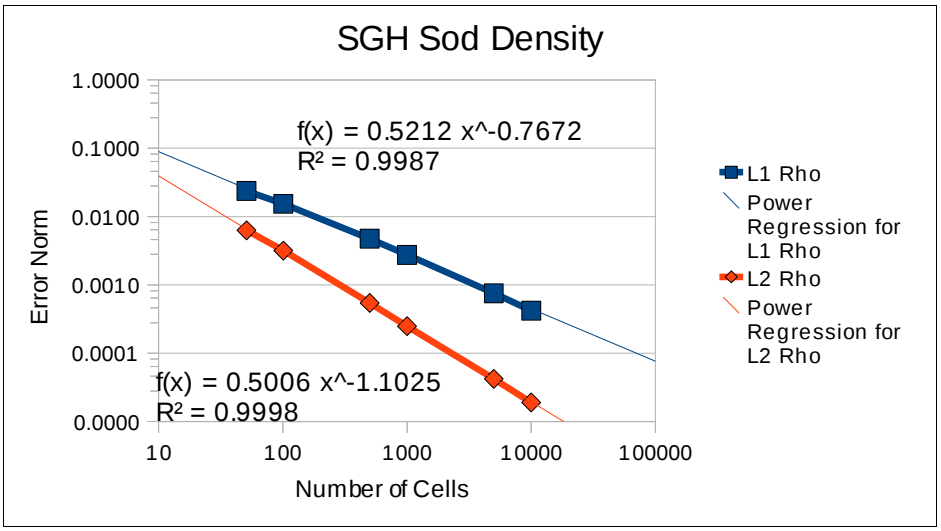

PCH Sod Pressure
f(x) = 1.1623 x^-0.8444
R² = 0.9999
f(x) = 0.7582 x^-1.1174
R² = 0.9997
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

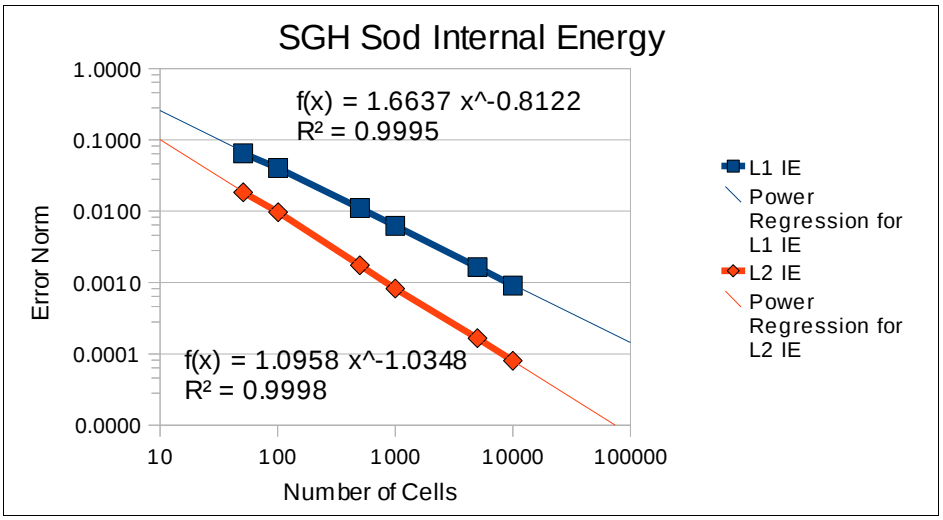

PCH Sod Density
f(x) = 0.8217 x^-0.8112
R² = 0.9997
f(x) = 0.5526 x^-1.0903
R² = 0.9999
Error Norm
Number of Cells
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


PCH Sod Internal Energy
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
f(x) = 3.1700 x^-0.8755
R² = 0.9988
f(x) = 1.7875 x^-1.0642
R² = 0.9963
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH

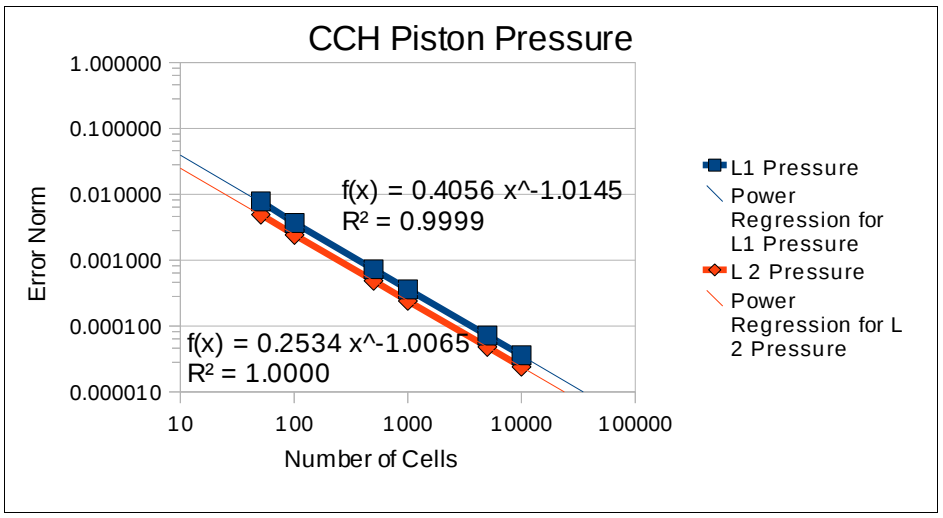

SGH Sod Pressure
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
f(x) = 0.6336 x^-0.7926
R² = 0.9993
f(x) = 0.6956 x^-1.1439
R² = 0.9998
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

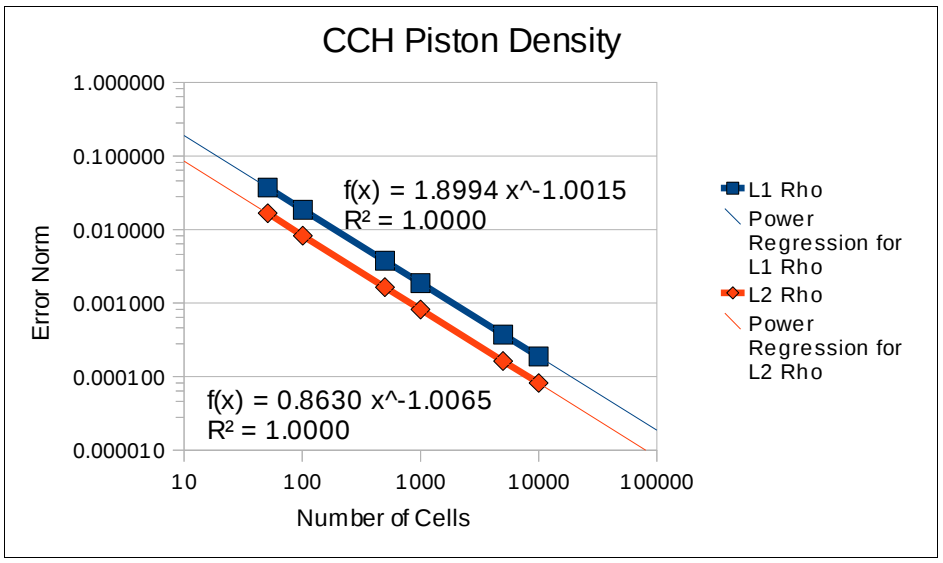

SGH Sod Density
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
f(x) = 0.5212 x^-0.7672
R² = 0.9987
f(x) = 0.5006 x^-1.1025
R² = 0.9998
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho

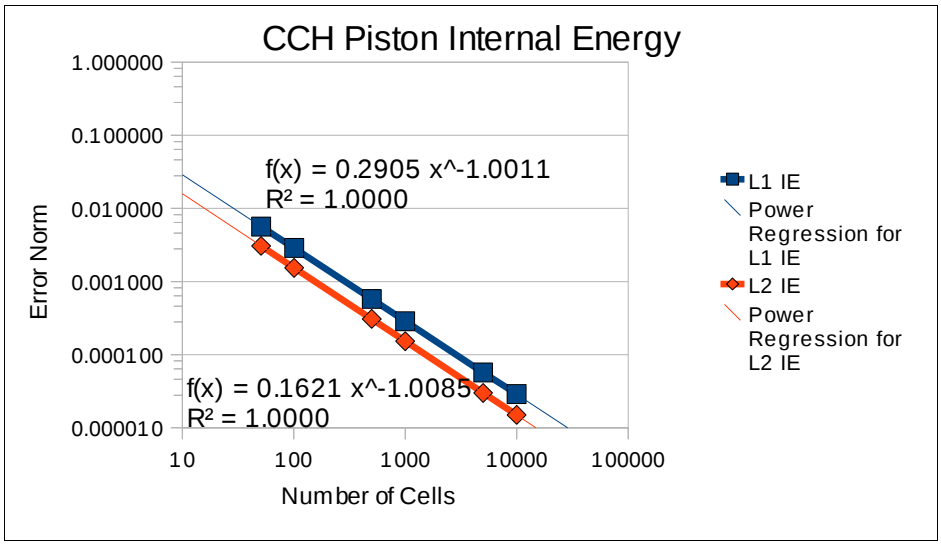

SGH Sod Internal Energy
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
f(x) = 1.6637 x^-0.8122
R² = 0.9995
f(x) = 1.0958 x^-1.0348
R² = 0.9998
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

PISTON

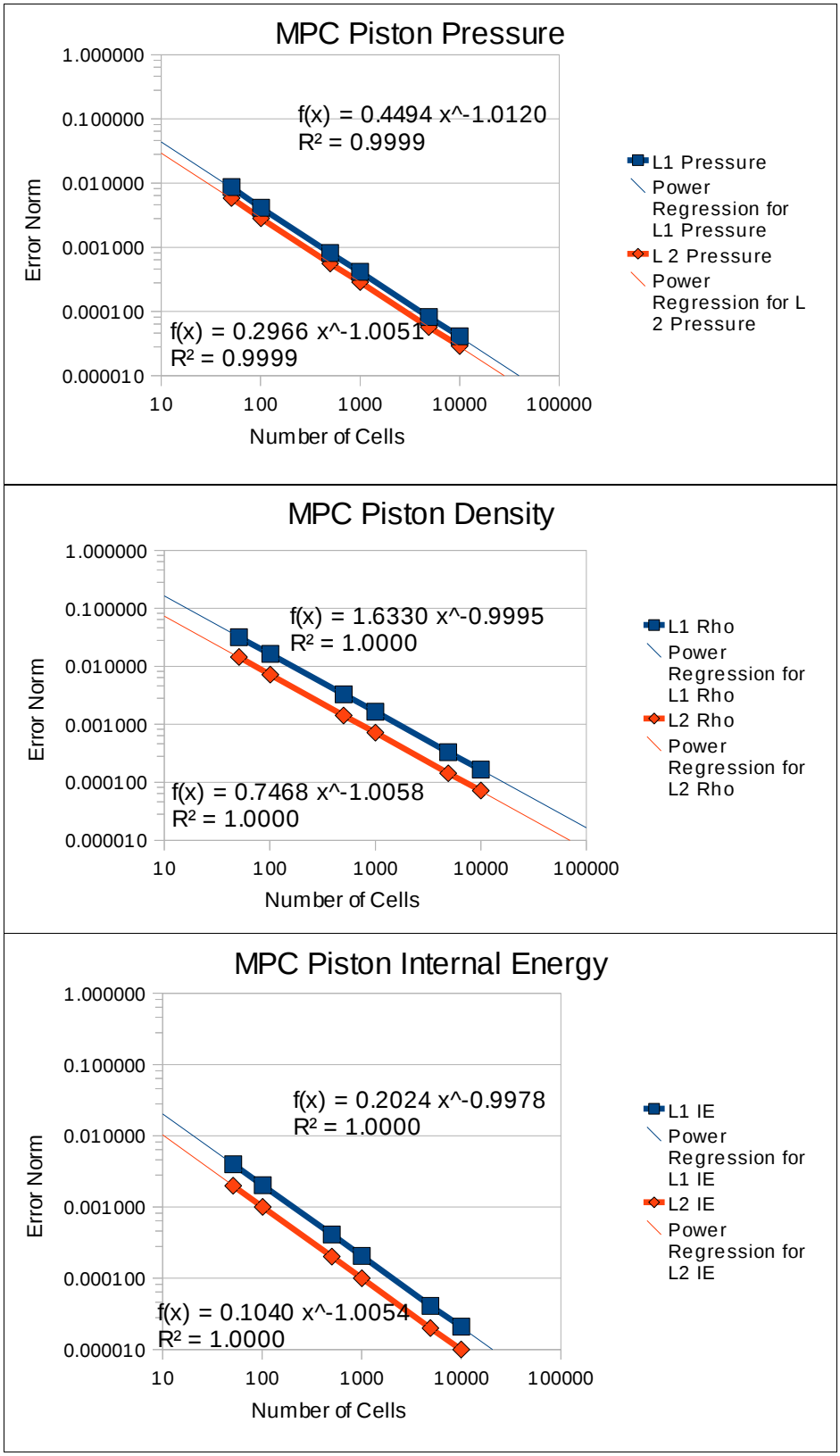
CCH


CCH Piston Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.4056 x^-1.0145
R² = 0.9999
f(x) = 0.2534 x^-1.0065
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

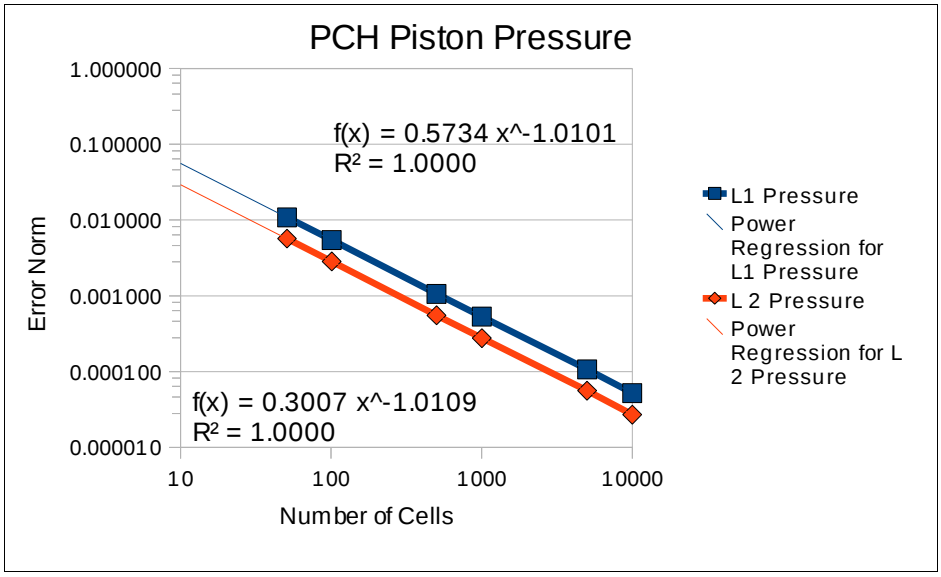

CCH Piston Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 1.8994 x^-1.0015
R² = 1.0000
f(x) = 0.8630 x^-1.0065
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho

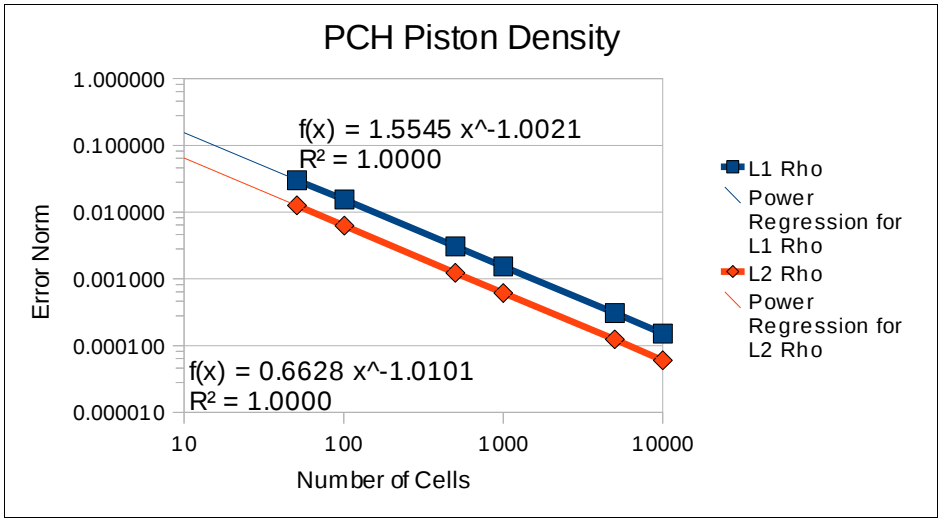

CCH Piston Internal Energy
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.2905 x^-1.0011
R² = 1.0000
f(x) = 0.1621 x^-1.0085
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

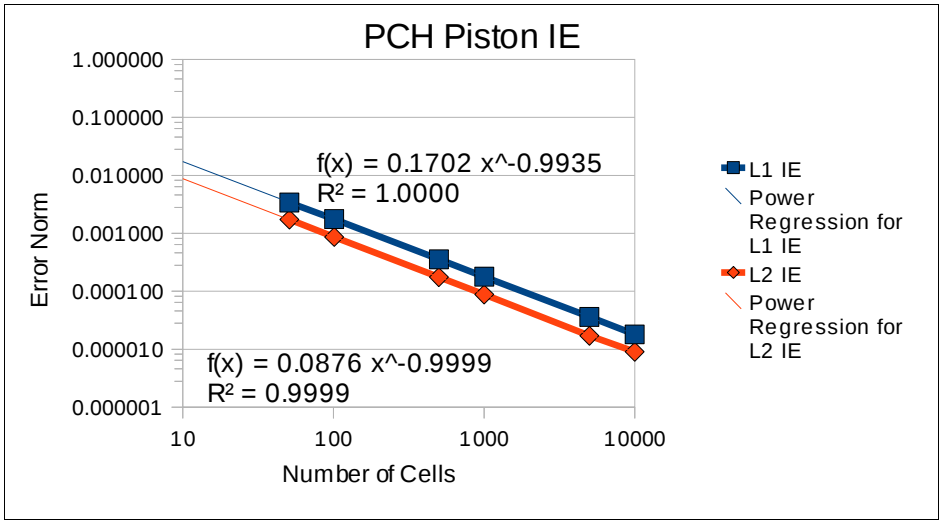
MPC


MPC Piston Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.4494 x^-1.0120
R² = 0.9999
f(x) = 0.2966 x^-1.0051
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure
MPC Piston Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 1.6330 x^-0.9995
R² = 1.0000
f(x) = 0.7468 x^-1.0058
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho
MPC Piston Internal Energy
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.2024 x^-0.9978
R² = 1.0000
f(x) = 0.1040 x^-1.0054
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

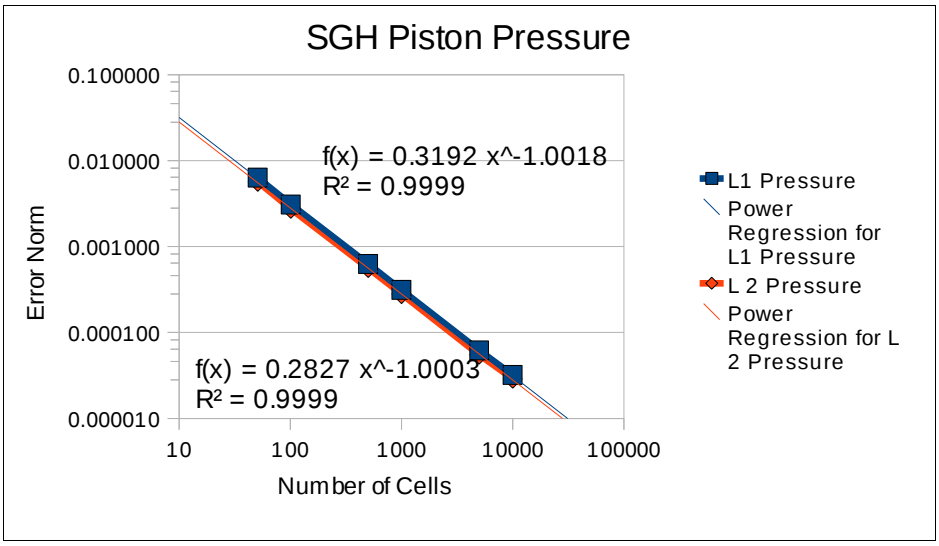
PCH

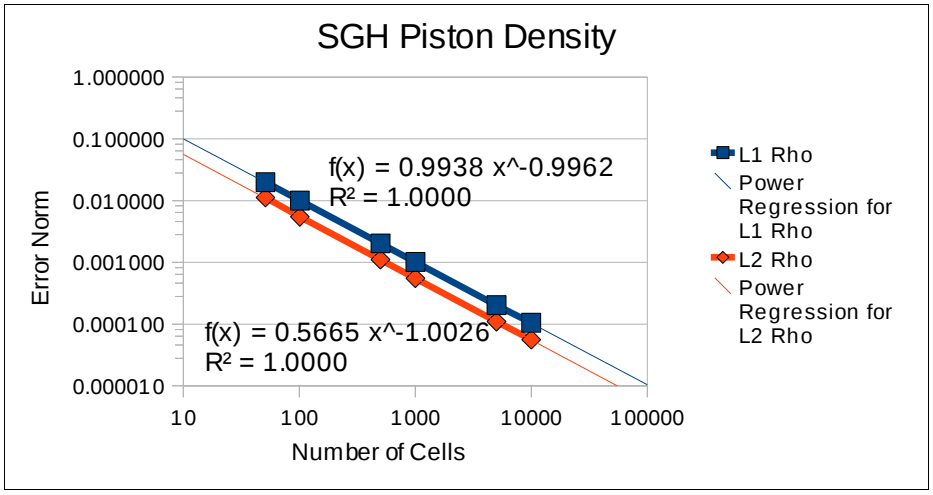

PCH Piston Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.5734 x^-1.0101
R² = 1.0000
f(x) = 0.3007 x^-1.0109
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

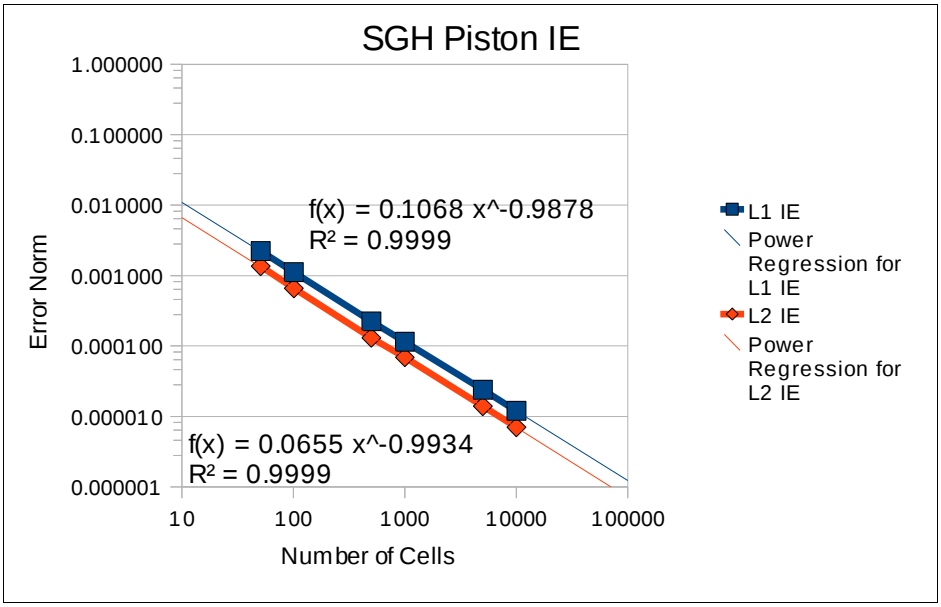

PCH Piston Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 1.5545 x^-1.0021
R² = 1.0000
f(x) = 0.6628 x^-1.0101
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


PCH Piston IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
f(x) = 0.1702 x^-0.9935
R² = 1.0000
f(x) = 0.0876 x^-0.9999
R² = 0.9999
10
100
1000
10000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH

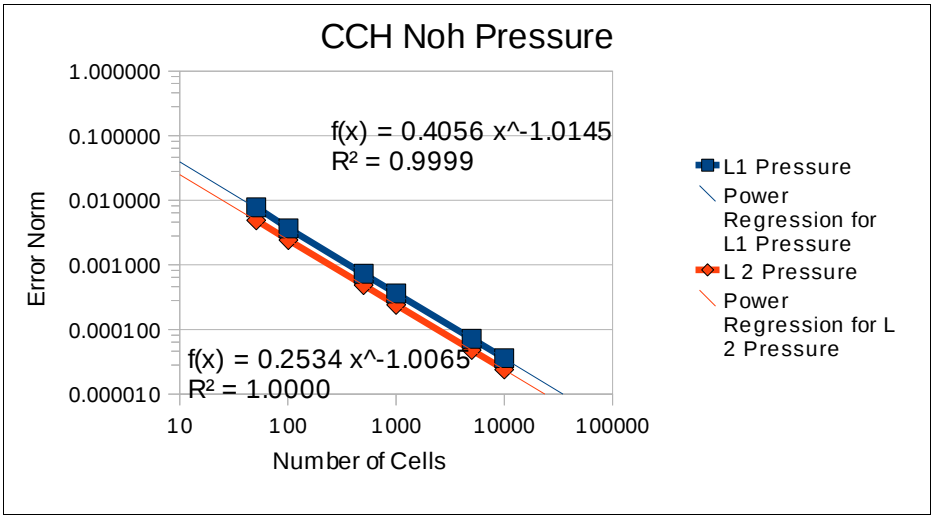

SGH Piston Pressure
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.3192 x^-1.0018
R² = 0.9999
f(x) = 0.2827 x^-1.0003
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

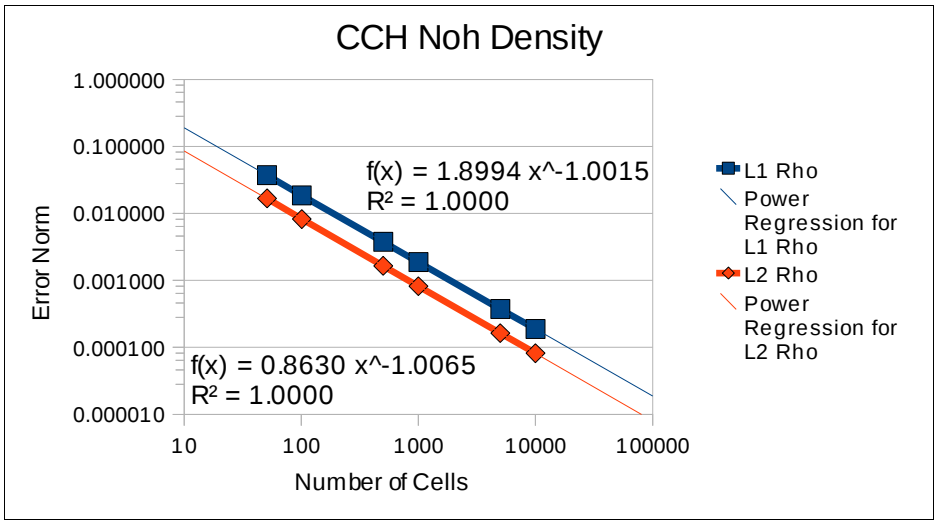

SGH Piston Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.9938 x^-0.9962
R² = 1.0000
f(x) = 0.5665 x^-1.0026
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho

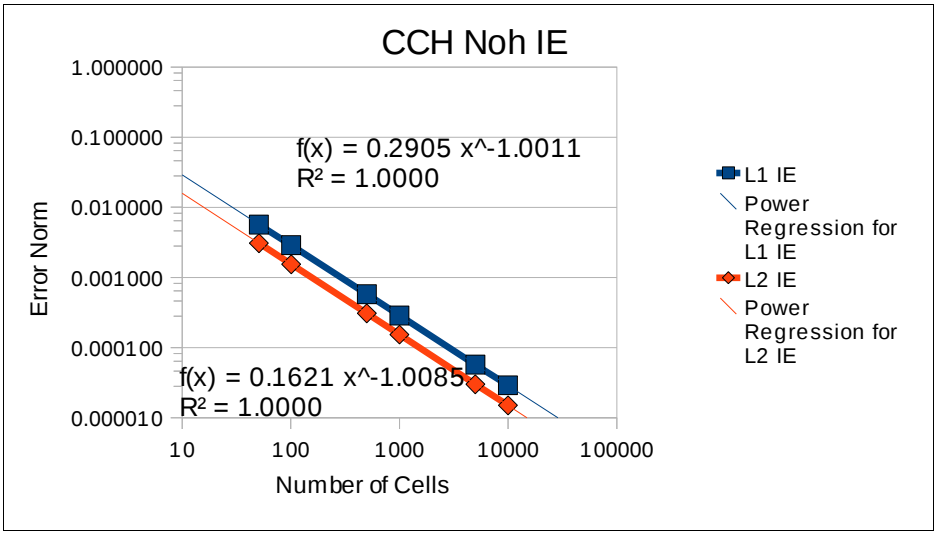

SGH Piston IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
f(x) = 0.1068 x^-0.9878
R² = 0.9999
f(x) = 0.0655 x^-0.9934
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

Noh Planar Coordinates

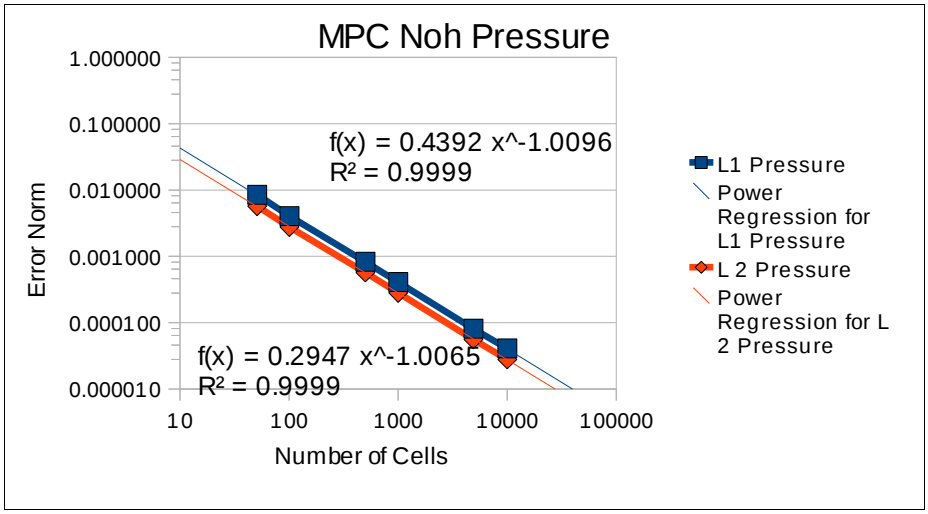
CCH

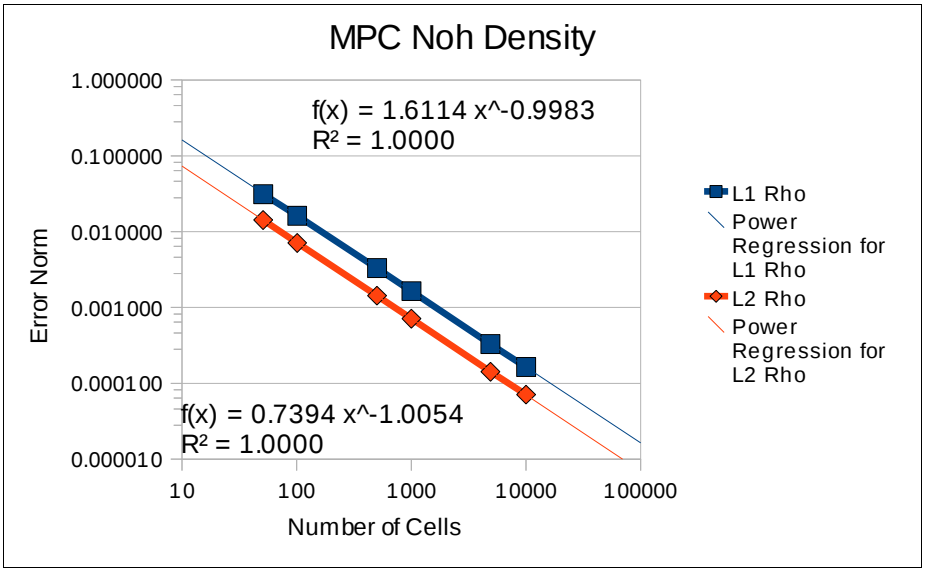


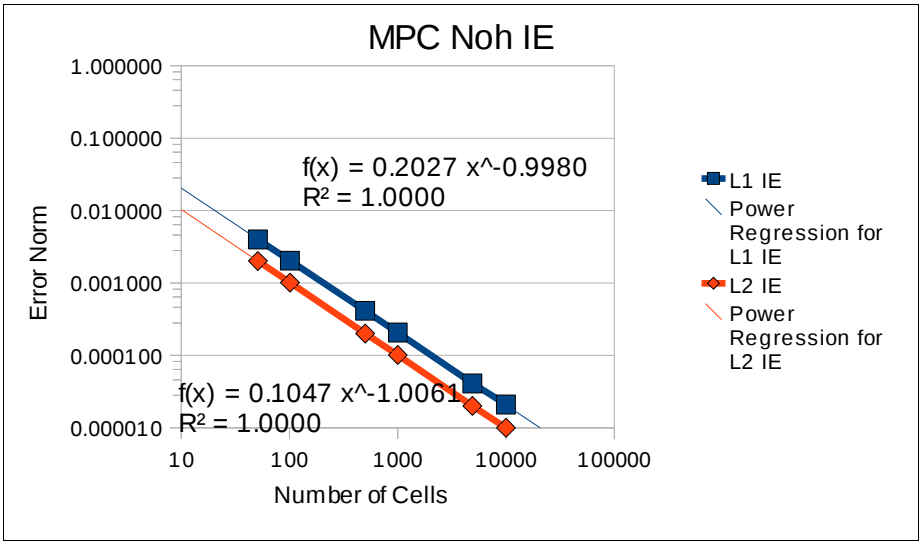




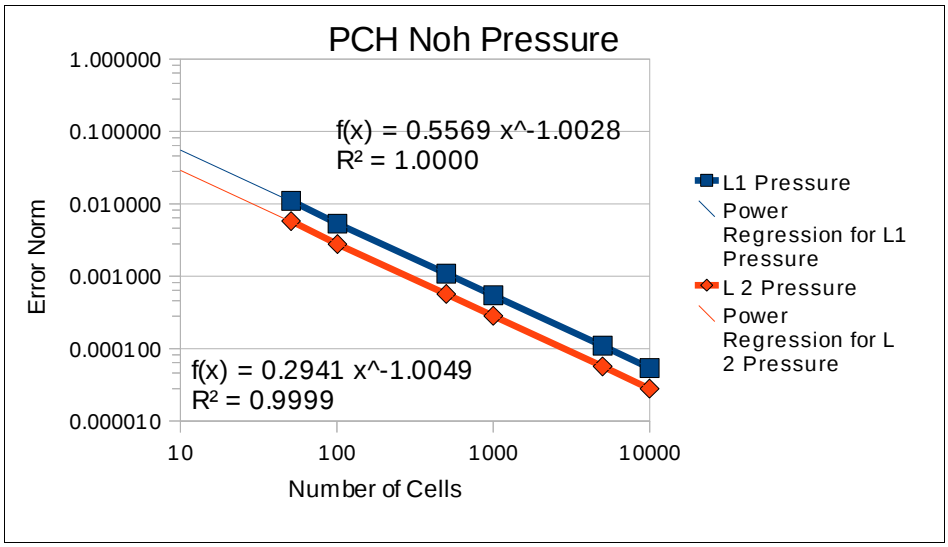
MPC

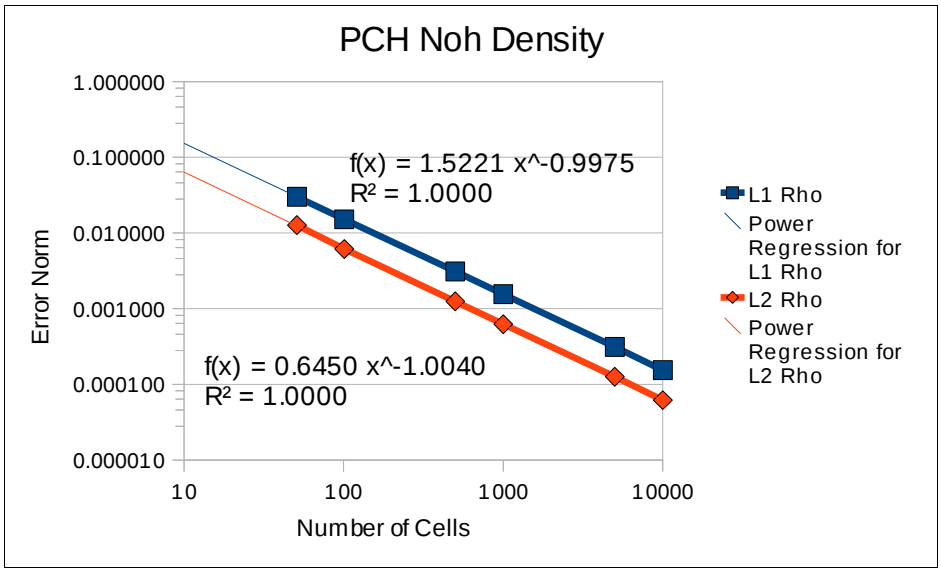

MPC Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.4392 x^-1.0096
R² = 0.9999
f(x) = 0.2947 x^-1.0065
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

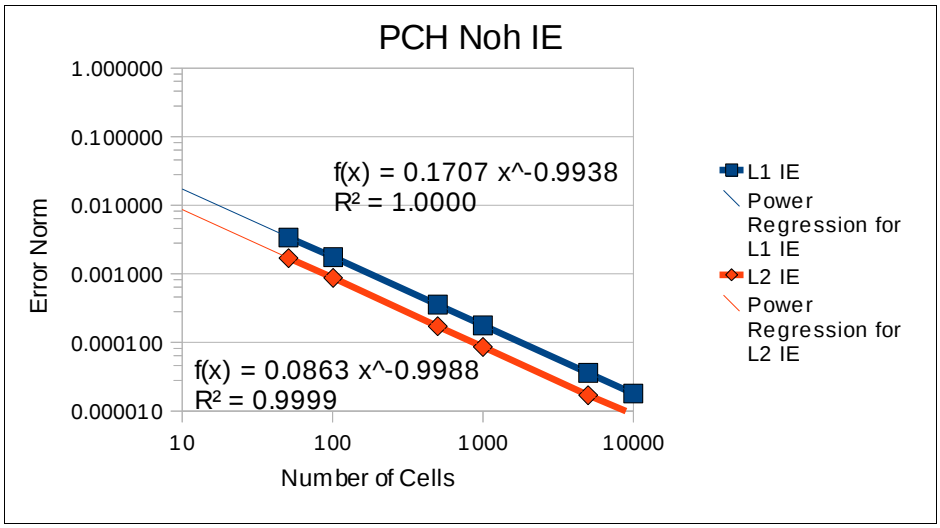

MPC Noh Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 1.6114 x^-0.9983
R² = 1.0000
f(x) = 0.7394 x^-1.0054
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


MPC Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.2027 x^-0.9980
R² = 1.0000
f(x) = 0.1047 x^-1.0061
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

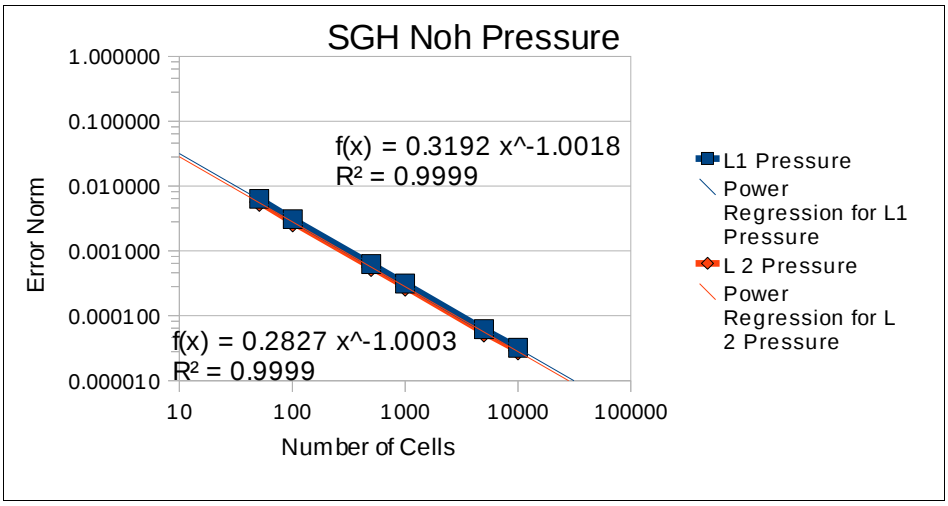
PCH

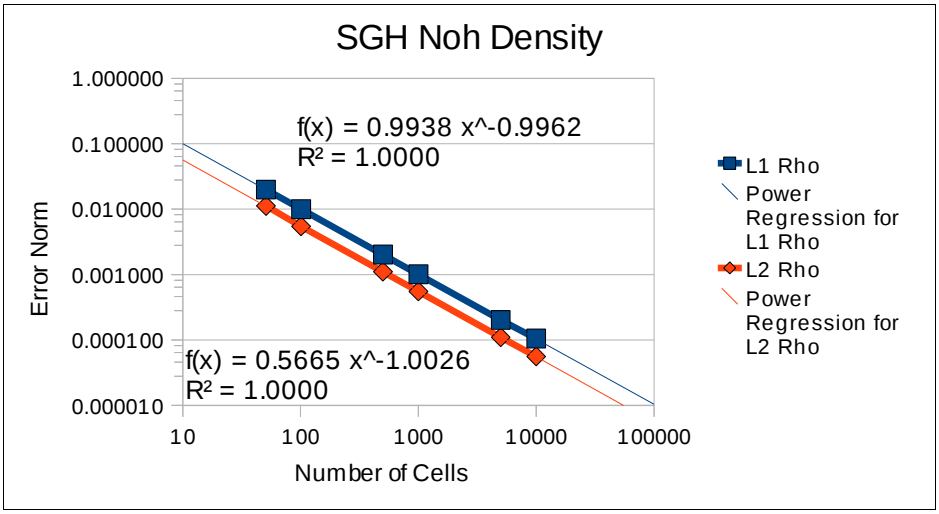

PCH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.5569 x^-1.0028
R² = 1.0000
f(x) = 0.2941 x^-1.0049
R² = 0.9999
10
100
1000
10000
Number of Cells
L1 Pressure
Power
Regression for L1
Pressure
L 2 Pressure
Power
Regression for L
2 Pressure

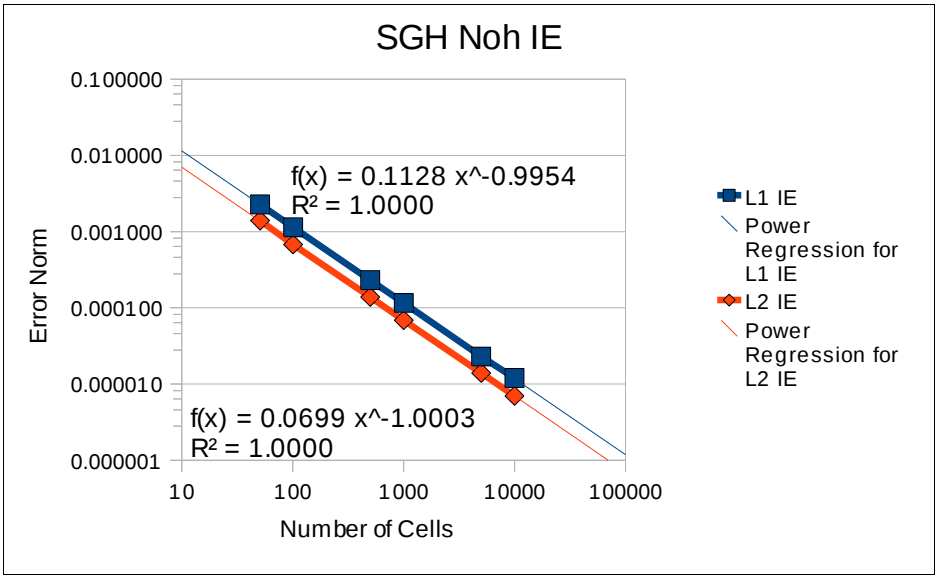

PCH Noh Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 1.5221 x^-0.9975
R² = 1.0000
f(x) = 0.6450 x^-1.0040
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 Rho
Power
Regression for
L1 Rho
L2 Rho
Power
Regression for
L2 Rho


PCH Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.1707 x^-0.9938
R² = 1.0000
f(x) = 0.0863 x^-0.9988
R² = 0.9999
10
100
1000
10000
Number of Cells
L1 IE
Power
Regression for
L1 IE
L2 IE
Power
Regression for
L2 IE

SGH

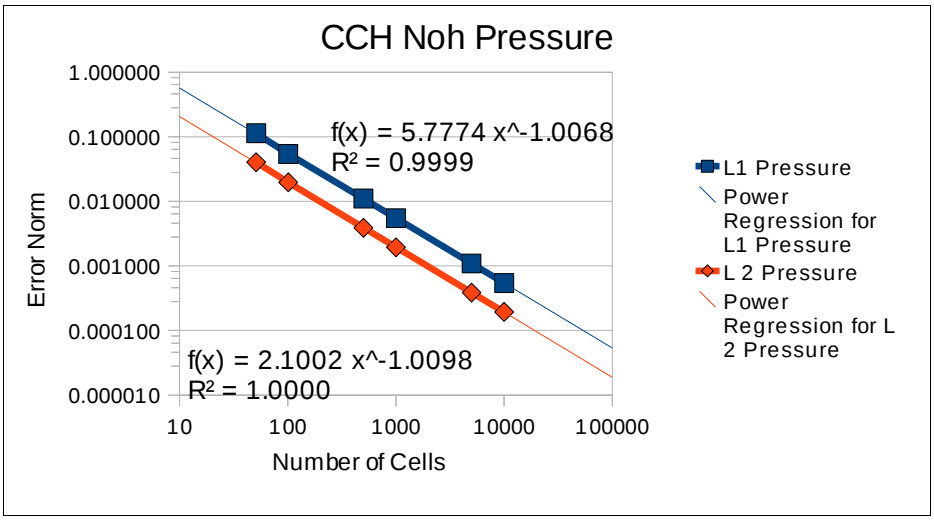

SGH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.3192 x^-1.0018
R² = 0.9999
f(x) = 0.2827 x^-1.0003
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power
Regression for L1
Pressure
L 2 Pressure
Power
Regression for L
2 Pressure

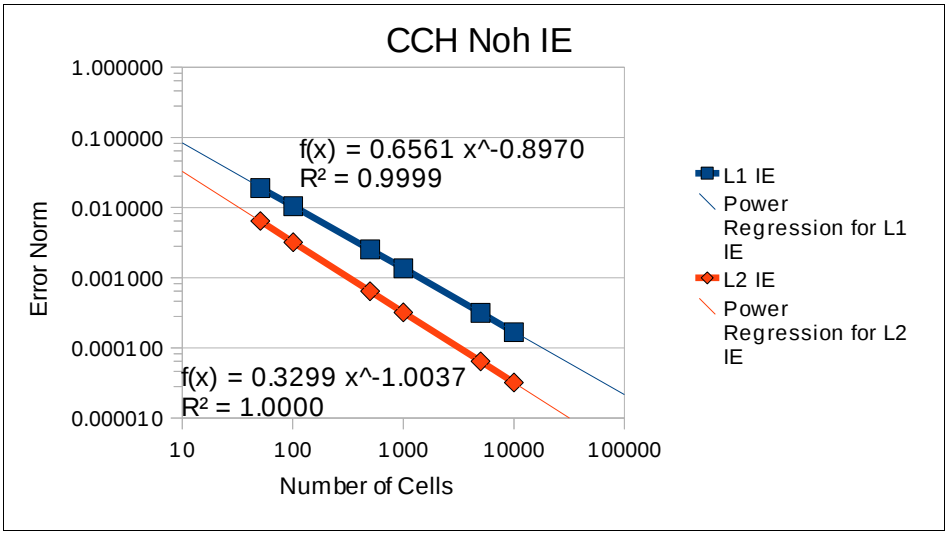

SGH Noh Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.9938 x^-0.9962
R² = 1.0000
f(x) = 0.5665 x^-1.0026
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 Rho
Power
Regression for
L1 Rho
L2 Rho
Power
Regression for
L2 Rho


SGH Noh IE
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
f(x) = 0.1128 x^-0.9954
R² = 1.0000
f(x) = 0.0699 x^-1.0003
R² = 1.0000
10
100
1000
10000
100000
Number of Cells
L1 IE
Power
Regression for
L1 IE
L2 IE
Power
Regression for
L2 IE

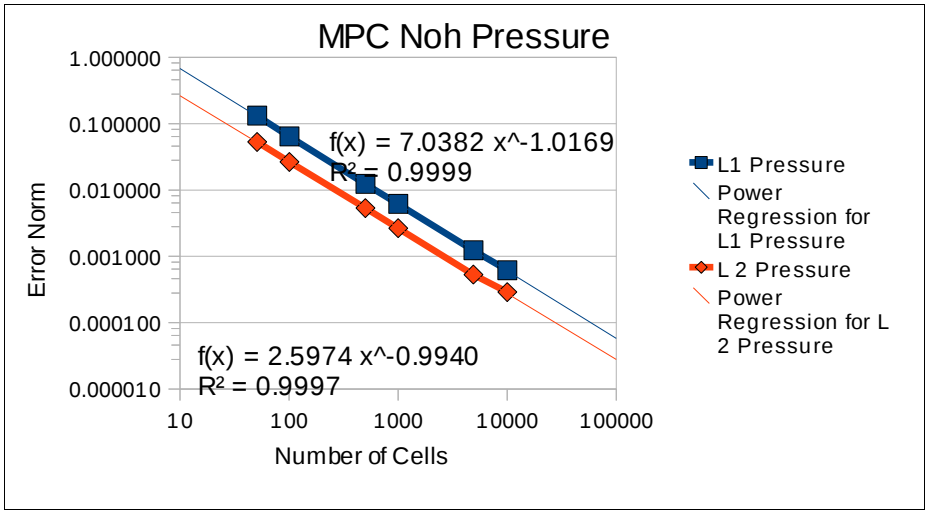
Noh Cylindrical

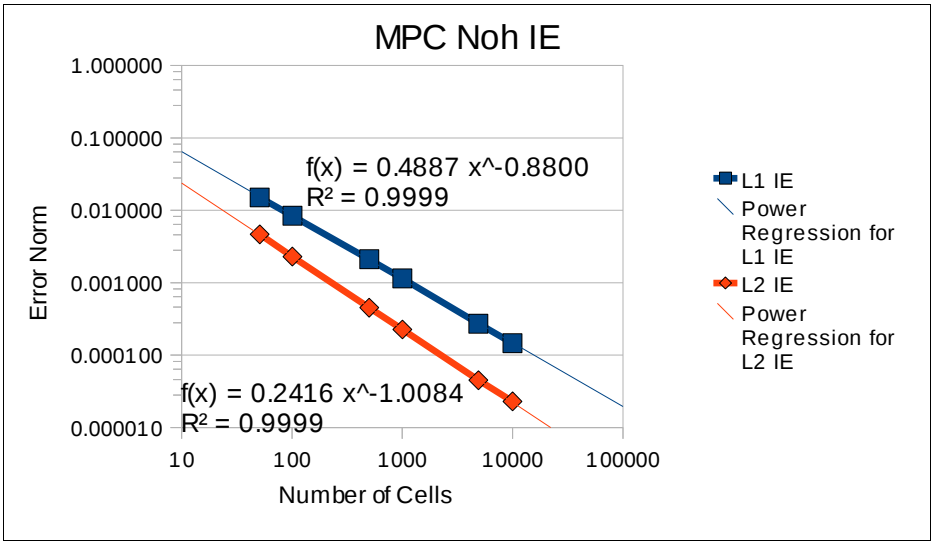
CCH



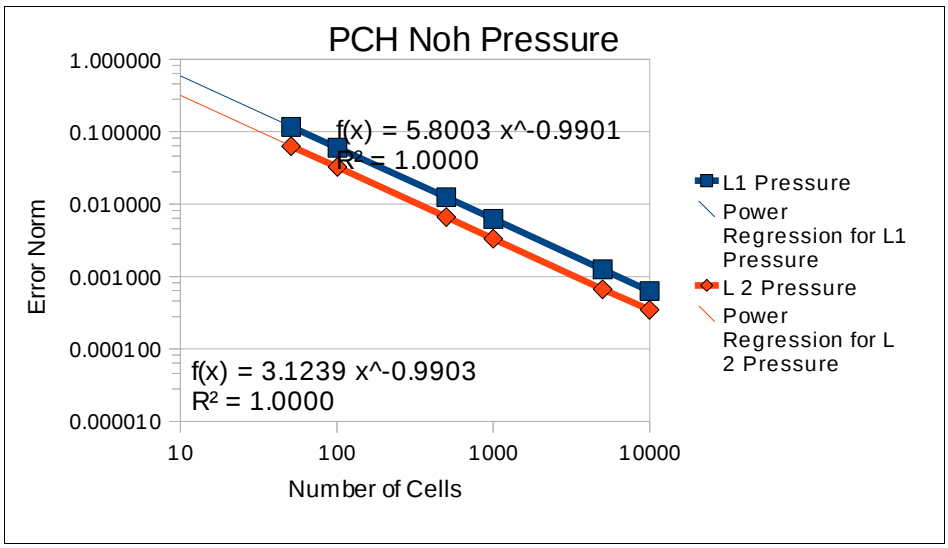


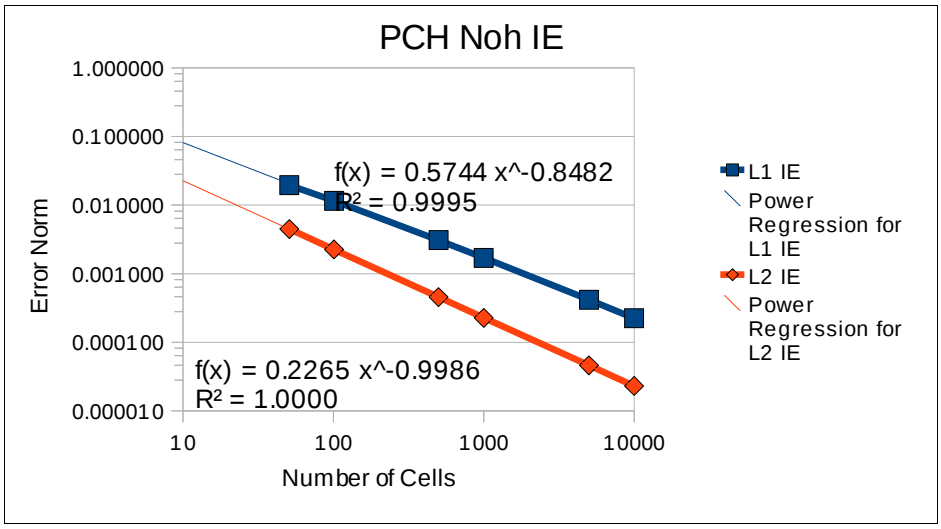
MPC


MPC Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 7.0382 x^-1.0169
R² = 0.9999
f(x) = 2.5974 x^-0.9940
R² = 0.9997
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

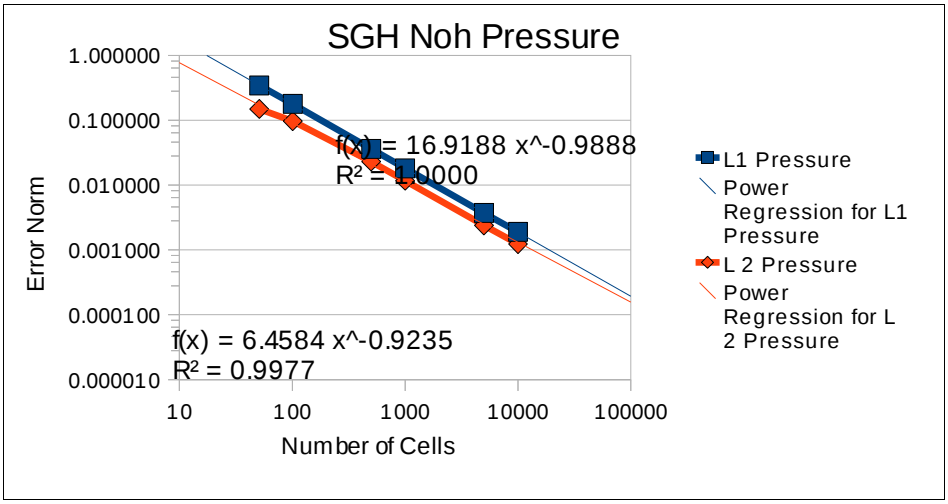

MPC Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.4887 x^-0.8800
R² = 0.9999
f(x) = 0.2416 x^-1.0084
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

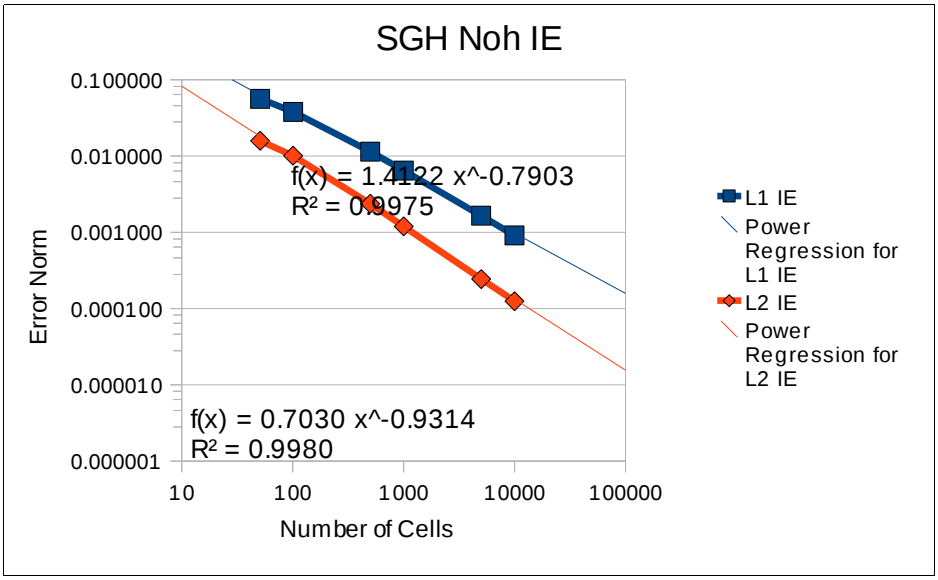
PCHA


PCH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 5.8003 x^-0.9901
R² = 1.0000
f(x) = 3.1239 x^-0.9903
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


PCH Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 0.5744 x^-0.8482
R² = 0.9995
f(x) = 0.2265 x^-0.9986
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

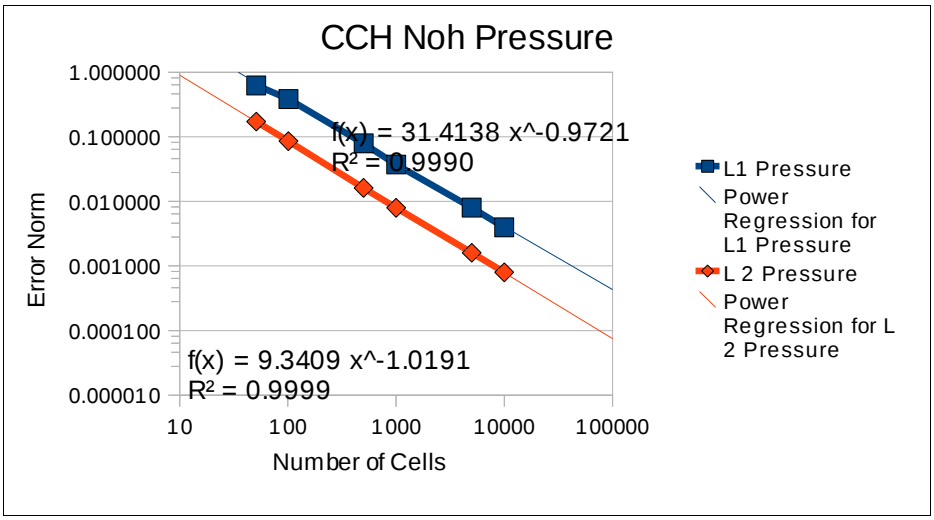
SGH

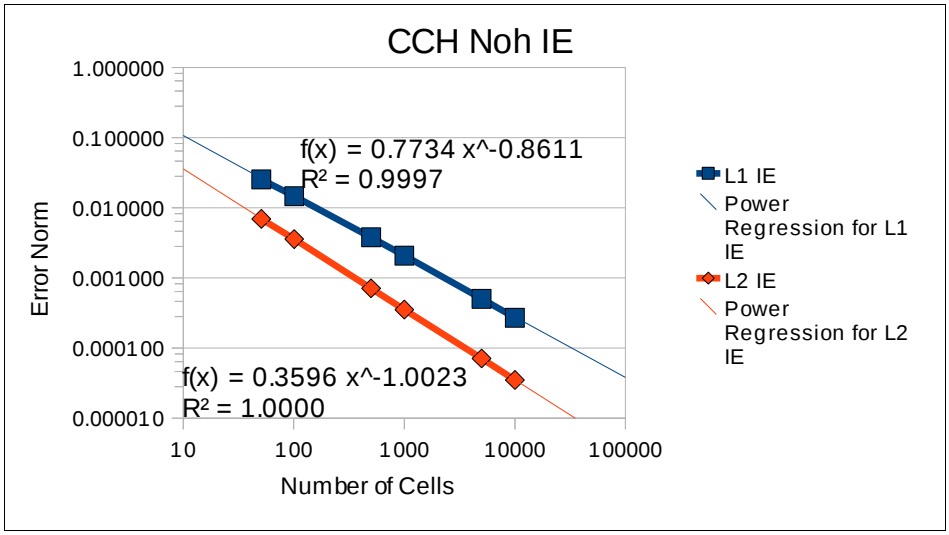

SGH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 16.9188 x^-0.9888
R² = 1.0000
f(x) = 6.4584 x^-0.9235
R² = 0.9977
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


SGH Noh IE
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
f(x) = 1.4122 x^-0.7903
R² = 0.9975
f(x) = 0.7030 x^-0.9314
R² = 0.9980
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

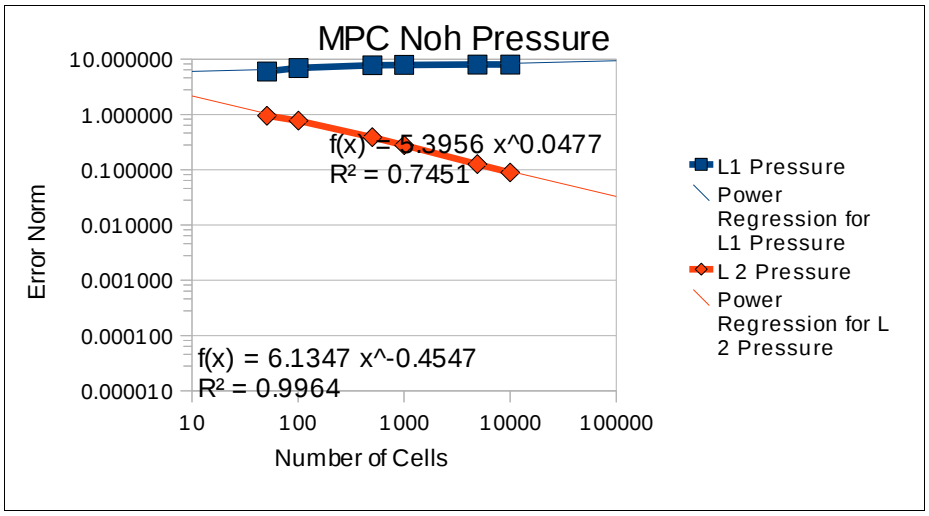
Noh Spherical

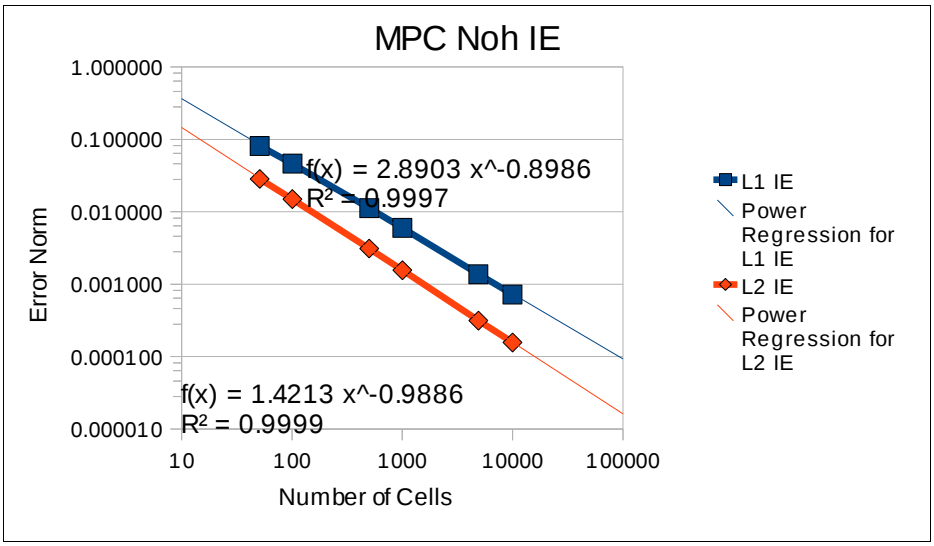
CCH



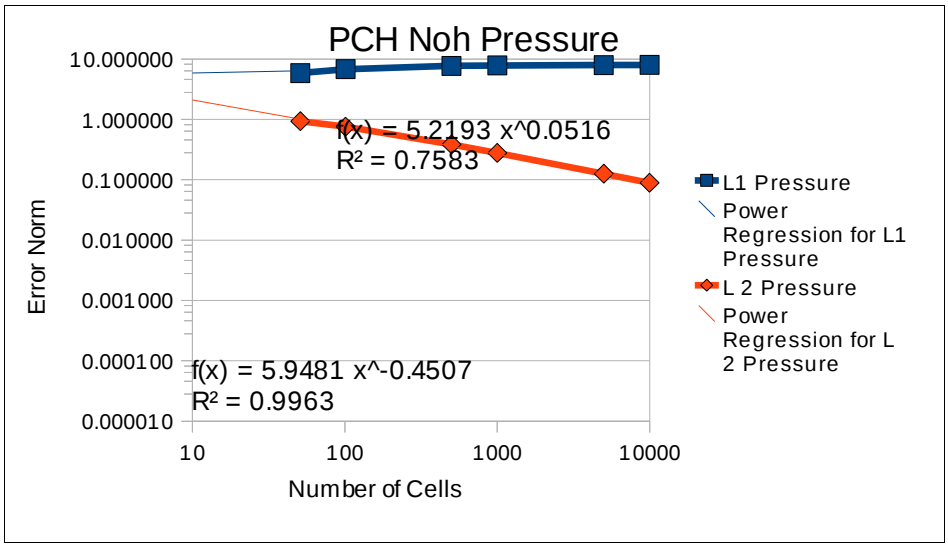


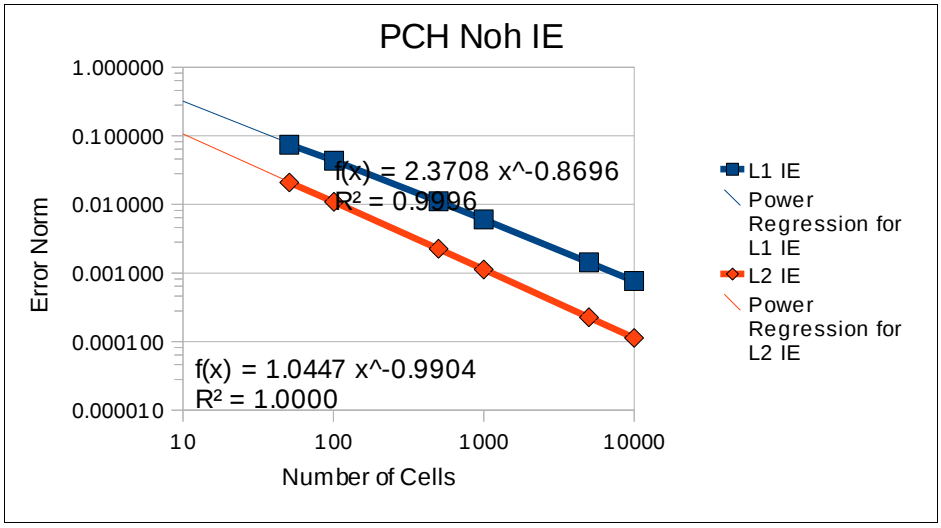
MPC


MPC Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 5.3956 x^0.0477
R² = 0.7451
f(x) = 6.1347 x^-0.4547
R² = 0.9964
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

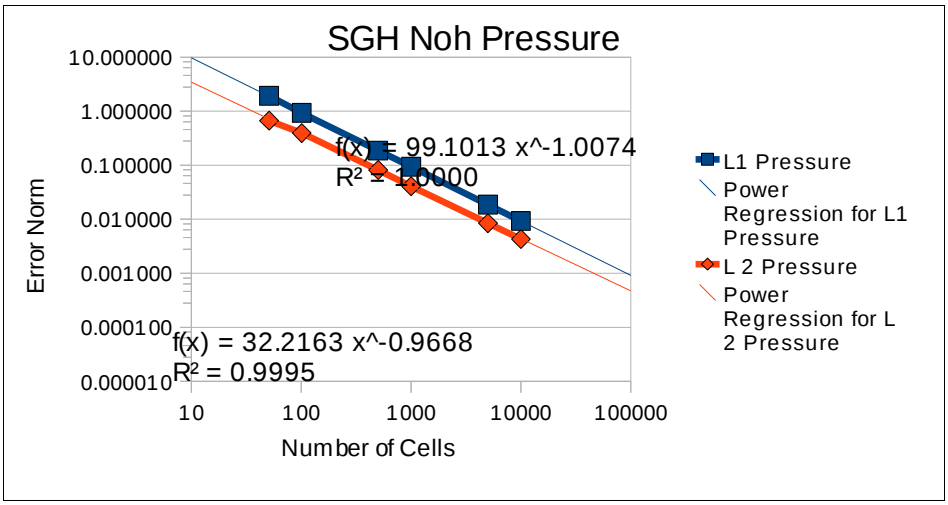

MPC Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 2.8903 x^-0.8986
R² = 0.9997
f(x) = 1.4213 x^-0.9886
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

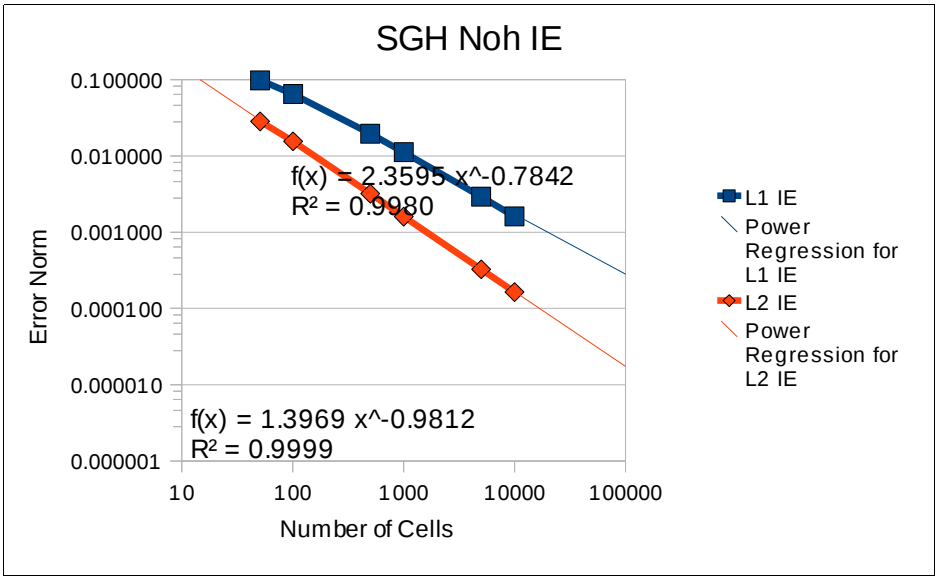
PCHA


PCH Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 5.2193 x^0.0516
R² = 0.7583
f(x) = 5.9481 x^-0.4507
R² = 0.9963
10
100
1000
10000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


PCH Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 2.3708 x^-0.8696
R² = 0.9996
f(x) = 1.0447 x^-0.9904
R² = 1.0000
10
100
1000
10000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH


SGH Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 99.1013 x^-1.0074
R² = 1.0000
f(x) = 32.2163 x^-0.9668
R² = 0.9995
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


SGH Noh IE
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
f(x) = 2.3595 x^-0.7842
R² = 0.9980
f(x) = 1.3969 x^-0.9812
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

Input Files

Sod Problem

```
$general
mats=2
np=101
L=100  (Radius in Cylindrical and Spherical)
init-ie?=1 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    (0: No, the point values will be initialized using the spacial variation
                        from user input.)
                    (1: Yes, the point values will be initialized using the average of the
                        cell values.)
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH   (CCH, SGH, PCH, or MPC [Modified PCH = Shifted CCH] )

(Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
                velocities)
             (1: Computes the density and total energy change with the Riemann velocities)
(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spatially averaged velocities)
NodePosOpt=0 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                at the end of the time integration loop)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=1.0
rho=1.0
ie=2.5
x1=0
x2=50
gamma=1.4
$end

$mat2
u=0.0
p=1.0
rho=0.125
ie=2.0
x1=50
x2=100
gamma=1.4
$end

$IO
dt0=0.0000000000001
tstop=20.1
dtdump=20
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Piston

```
(Note: Use parentheses to enclose comments)
$general
mats=1
np=51
L=1
init-ie?=0 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    1: Yes, the point values will be initialized using the average of the
cell values.
                    0: No, the point values will be initialized using the spacial variation
from user input.
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=1.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH

(3 Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                    at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                    The point_x values are updated the same as in NodePosOpt=0.
                    This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=0.0
rho=1.0
ie=0
x1=0
x2=1
gamma=1.66666666667
$end

$IO
dt0=0.0000000001
tstop=0.61
dtdump=0.6
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Noh

```
(Note: Use parentheses to enclose comments)
$general
mats=1
np=51
L=1
init-ie?=0 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    1: Yes, the point values will be initialized using the average of the
cell values.
```

```
                    0: No, the point values will be initialized using the spacial variation
from user input.
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=-1.0    (Boundary Velocity at end, used when fixed at right)
Method=CCH

(3 Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                    at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                    The point_x values are updated the same as in NodePosOpt=0.
                    This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=-1.0
p=0.0
rho=1.0
ie=0
x1=0
x2=1
gamma=1.66666666667
$end

$IO
dt0=0.0000000001
tstop=0.61
dtdump=0.6
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Sedov

```
(Note: Use parentheses to enclose comments)
$general
mats=2
np=61
L=1.2  (Radius in Cylindrical and Spherical)
init-ie?=1 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    (0: No, the point values will be initialized using the spacial variation
from user input.)
                    (1: Yes, the point values will be initialized using the average of the
                           cell values.)

BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH   (CCH, SGH, PCH, or MPC [Modified PCH = Shifted CCH] )

(3 Options for PCH)
```

```
VelOpt=1     (0: Computes the density and total energy change with the averaged point
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                    at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                    The point_x values are updated the same as in NodePosOpt=0.
                    This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=0.0
rho=1.0
ie=15  (Extensive IE=0.3 MBar cc  Volume=0.02 cc [that's 0.3/1])
x1=0
x2=0.02
gamma=1.66666666666667
$end

$mat2
u=0.0
p=0.0
rho=1.0
ie=0.0
x1=0.02
x2=1.2
gamma=1.66666666666667
$end

$IO
dt0=0.0000000000001
tstop=1.01
dtdump=1.0
CFL=0.2
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Selected Portions of Source Code

CCH Riemann Solver:

```
avg=(cell_u[i-1]+cell_u[i])/2.0;

                    c=sqrt(cell_gam[i-1]*cell_p[i-1]/cell_rho[i-1]);     // c for cell on
left
                    if (c<0.000000001 || c!=c) { c=0.000000001; }       //If c<1e-9 or
c=nan,set a minimum value for c

                    /* Calculate Compression-Aware mu on the left */
                    if ((cell_u[i]-cell_u[i-1])>0)
                    {
                          /* Expansion */
                          point_mu[0]=cell_rho[i-1]*c;
```

```
                }
                else
                {
                        /* Compression */
                        point_mu[0]=cell_rho[i-1]*c+cell_rho[i-1]*
                                ((cell_gam[i-1]+1.0)/2.0)*fabs(avg-cell_u[i-1]); //mu on
                                                                                        left
                }


                c=sqrt(cell_gam[i]*cell_p[i]/cell_rho[i]);    // c for cell on right
                if (c<0.000000001 || c!=c) { c=0.000000001; } //If c<1e-9 or c=nan,set a
                                                                minimum value for c

                /* Calculate Compression-Aware mu on the right */
                if ((cell_u[i]-cell_u[i-1])>0)
                {
                        /* Expansion */
                        point_mu[1]=cell_rho[i]*c;
                }
                else
                {
                        /* Compression */
                        point_mu[1]=cell_rho[i]*c+cell_rho[i]*
                                ((cell_gam[i]+1.0)/2.0)*fabs(avg-cell_u[i]); //mu on right
                }

                /* Compute u* at each point */
                point_u[i]=(cell_p[i-1]*point_normal[0]+cell_p[i]*
                        point_normal[1]+point_mu[0]*cell_u[i-1]
                        +point_mu[1]*cell_u[i])/(point_mu[0]
                        +point_mu[1]);

                /* Compute p* at each point */
                point_p[i]=point_mu[1]*(point_u[i]-cell_u[i])-cell_p[i]*
                        point_normal[1];

                /* Check to verify Riemann Pressures are equal and opposite */
                point_pstar_check=point_mu[0]*(point_u[i]-cell_u[i-1])-cell_p[i-1]*
                        point_normal[0];
                if (fabs(point_p[i]+point_pstar_check)>pstar_tol)
                {       check++;     }
```

Runge-Kutta Time Integration Solution Loop:

```
do
        {
                alpha=1.0/(nstage+1.0-istage);  // Establish the coefficient
                CCHSolveRHS(mass,alpha,SphAvgOpt);
                istage++;    // increase the RK cycle number

        } while (istage<=nstage);
```

CCH Conservation Equations:

```
for (i=0;i<nz;i++)
        {
                if (PiOpt==1) Area=GetAreaCCH(i);
```

```
            else if (PiOpt==0) Area=GetAreaCCH_NoPi(i);

            /* Calculate Forces on either side of the control volume */
            Fstar[0]=-point_p[i]*point_normal[1]*Area;    // Force on Left side of CV
            Fstar[1]=-point_p[i+1]*point_normal[0]*Area;  // Force on Right side of CV

            v[i]=(1.0/mass[i])*(Fstar[0]+Fstar[1]);
            e[i]=(1.0/mass[i])*(Fstar[0]*point_u[i]+Fstar[1]*point_u[i+1]);
      }
/*  +++++++++++++++End Calculate RHS ++++++++++++++++++++++++++++++ */
/* +++++++++++++++Begin Time Step Forward ++++++++++++++++++++++++++ */
      for (i=0;i<np;i++)
      {
            /* Calculate new point_x values */
            point_x[i]=point_x0[i]+alpha*dt*point_u[i];
      }
      for (i=0;i<nz;i++)
      {
            /* Calculate new dx */
            cell_dx[i]=point_x[i+1]-point_x[i];

            /* Calculate new rho */
            if (PiOpt==1) Volume=GetVolumeCCH(i);
            else if (PiOpt==0) Volume=GetVolumeCCH_NoPi(i);
            cell_rho[i]=mass[i]/Volume;

            /* Calculate new u */
            cell_u[i]=cell_u0[i]+alpha*dt*v[i];

            /* Calculate new total energy */
            cell_te[i]=cell_te0[i]+alpha*dt*e[i];

            /* Calculate new internal energy */
            cell_ie[i]=cell_te[i]-0.5*pow(cell_u[i],2.0);

            /* Calculate new pressure using EOS-Gamma Law Gas */
            cell_p[i]=cell_rho[i]*(cell_gam[i]-1.0)*cell_ie[i];
      }
```

Plasma Mixing in ICF Applications

(Erik Vold, mentor)

Plasma Mixing In ICF Applications

Daniel Fenn, Ryan Moll, Erik Vold

August 19, 2013

Abstract

Accurate computer modeling of inertial confinement fusion processes represents a current challenge in ICF research. Current computer simulations tend to overestimate the yield of fusion reactions initiated in laboratory experiments, suggesting a need for the implementation of more sophisticated physical models. Small scale, atomic-level mixing may occur, possibly having a significant impact on fusion reactions in these experiments. Here we model thermal conduction, viscosity, and mass diffusion as diffusion processes in two 1D, Lagrangian hydro codes in spherical geometry. We will show that thermal conduction has the greatest effect on peak temperatures in our simulated ICF implosions; however viscosity and mass diffusion are potentially significant as well. We will also draw contrasts between our results and the results of simulations done with a 1D Lagrangian hydro code called Helios, and compare the results of our two independently developed codes.

1 Introduction

Intertial confinement fusion (ICF) is one of two major approaches to acheiving nuclear fusion as a source of sustainable energy. The ICF approach utilizes a spherical capsule of initial radius typically less than 1 mm, filled with a deuterium-tritium gas. We restrict our attention here to direct-drive laser systems, in which high-energy laser beams illuminate the entire surface of the fuel pellet. This applied laser energy results in implosion of the capsule, and the resulting shock wave that propagates to the center has the effect of increasing densities and temperatures in the central region to the point of initiating a nuclear fusion reaction in the DT fuel.

Current computer simulations of the ICF process overestimate the yield of the resulting nuclear fusion reaction, implying that there are important physical effects

that are not being accounted for in the computer models [1]. Fluid mixing is thought to be the primary source of this discrepancy. One possible source of mixing arises from large-scale fluid instabilities that develop inside the capsule during the implosion. Laser power is not distributed uniformly over the surface of the capsule, potentially leading to the growth of Rayleigh-Taylor and Richtmyer-Meshkov instabilities and turbulence in the interior of the capsule. In addition to this large-scale mixing, atomic-level mixing is thought to occur by means of plasma transport processes.

It is common for ICF simulations to implement phenomenological models of mixing; however, it appears that these models do not adequately describe atomic-scale mixing [2]. We explore here a different approach, in which plasma kinetic effects are modeled in a simple approximation by diffusion. We implement diffusion of momentum, temperature, and mass to simulate the effects of viscosity, heat conduction, and diffusive mixing of shell and fuel materials in a straightforward 1D Lagrangian-frame, spherically-symmetric model.

To facilitate the process of code verification, each of us developed an entire code, independently of the other. Throughout this report, we will indicate important similarities and differences between the codes, and highlight the benefits of this approach.

2 Governing Equations

The ICF system was modeled using the Euler equations for a binary fluid. Included are equations describing the evolution of density, velocity, temperature, pressure, and species number fraction. For all of our simulations, an ideal gas equation of state was used for closure of the system.

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u}$$

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla : (\eta_0 \nabla \mathbf{u})$$

$$\frac{N}{(\gamma - 1)\, V} \frac{DT}{Dt} = -p \nabla \cdot \mathbf{u} + \nabla \cdot (\kappa_e \nabla T)$$

$$\frac{DC}{Dt} = \nabla \cdot (D_{12} \nabla C)$$

$$N = \rho V N_{av} \left(\frac{C}{m_{DD}} + \frac{(1 - C)}{m_{CH}} \right)$$

$$p = \frac{N}{V} kT$$

In the above equations,ρ,$\mathbf{u}$,T,C and p represent density, velocity, temperature, species mass fraction and pressure, respectively. Also, N is the number of particles, N_{av} is Avogadro's number, V is the zone volume, and γ is the adiabatic index ($\gamma = \frac{5}{3}$ for all simulations). The plasma coefficients for thermal conduction, κ_e, and viscosity, η_0, are defined as [3]

$$\eta_0 = 2.006 \times 10^7 \frac{kT_i^{5/2}}{\lambda} \mu^{1/2} \frac{\text{g}}{\text{cm} \cdot \text{s}}$$
$$\kappa_e = 1.101 \times 10^6 \frac{kT_e^{5/2}}{\lambda m_e} \frac{1}{\text{cm} \cdot \text{s}},$$

where $k = 1.6 \times 10^{-12} \frac{\text{erg}}{\text{eV}}$ is Boltzmann's constant, $\lambda \approx 5$ is the Coulomb logarithm, $m_e$is the electron mass in grams, and μ is the mean particle mass in proton masses. The variables T_i and T_e are the ion temperature and the electron temperature, respectively. It is important to note that throughout all of our simulations we make the approximation that $T_i = T_e = T$. The two fluids considered in our simulations were fuel (DT) and shell material (CH). The mass diffusion coefficient describes the diffusion of species one (fuel) into species two (shell material) and is defined using the following prescription

$$D_{12} = 2470 \frac{(kT_1)^{5/2}}{m_1 \lambda} \left(2.4 \frac{\rho_1}{m_1} + \frac{\rho_2 z_2^2}{m_2} \right)^{-1},$$

where $m_{1,2}$ is the mass of species 1 or 2 in atomic mass units. The atomic number of species 2 is z_2 and species 1 is assumed to be hydrogen. Also, $\rho_{1,2}$ are the densities of species 1 and species 2 in the zone

For preliminary testing, all diffusion terms were set to zero. In order to add stability to the test system, particularly in the vicinity of the shock, a very basic prescription for artificial viscosity was also used (artificial viscosity is displayed in its discretized form and i is a spatial index)

$$q_i = \begin{cases} \rho_i |\Delta u_i| \left(c_Q |\Delta u_i| + c_L c_{s,i} \right) & \Delta u_i < 0 \\ 0 & \text{otherwise} \end{cases},$$

where $\Delta u = u_i - u_{i+1}$, $c_Q = \frac{1}{4}(\gamma + 1)$, and $c_L = \frac{1}{2}$. The sound speed, c_s, is defined in the following way

$$c_s = \sqrt{\frac{\gamma P}{\rho}}.$$

The conditional statement above for q implies that artificial viscosity is only active in zones that are being compressed. It is interesting to note that even after turning diffusion terms on, it was still necessary in one of our codes to keep artificial viscosity to maintain numerical stability.

The above equations were those used in Ryan's code. Daniel's code was very similar, with a few notable differences. In his code, C was defined as a species number fraction, causing N to be defined as

$$N = \rho V \left(\frac{1}{C m_{DD} + (1 - C) m_{CH}} \right),$$

where the masses are given in grams per particle. Additionally, Daniel's code used an artificial viscosity given by [4]

$$q_i = \begin{cases} \rho_i |\Delta u_i| \left(c_Q |\Delta u_i| \left(1 - \phi^2\right) + c_L c_{s,i} \left(1 - \phi\right) \right) & \Delta u_i < 0 \\ 0 & \text{otherwise} \end{cases},$$

$$\text{where } \phi_i = \max\left(0, \min\left(1, 2R_{i+1}, 2R_{i-1}, \frac{1}{2}\left(R_{i+1} + R_{i-1}\right)\right)\right)$$

$$\text{and } R_{i\pm1} = \frac{\Delta u_{i\pm1}}{\Delta u_i}.$$

3 Numerical Method

ICF implosions were simulated using two different 1-D Lagrangian hydro codes. In the early stages of development one code was written using spherical geometry and the other used planar geometry. Later on, both of our codes were adapted to use exclusively spherical geometry. Also, for preliminary verification purposes the governing equations were first solved without the diffusion terms. To discretize the governing equations we used a first order finite difference method. To implement the discretized equations into a simulation code we used a solution method similar to that prescribed by Pember & Anderson [4]. In what follows, p represents the thermodynamic pressure with the artificial viscosity added to it, unless otherwise noted. The basic structure of our codes was the following (note that the superscript n denotes a temporal index, and the subscript i denotes a spatial index):

1. Solve for the new velocity

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\rho \Delta r} \left(p_i^n - p_{i-1}^n \right)$$

2. Use new velocities to determine new zone boundary positions

$$r_i^{n+1} = r_i^n - \frac{1}{2}\Delta t \left(u_i^{n+1} + u_i^n\right)$$

3. Using the new boundary positions, new zone volumes can be calculated, and new densities can be calculated in a way that ensures mass conservation

$$\rho_i^{n+1} = \rho_i^n \frac{V_i^n}{V_i^{n+1}}$$

4. New temperatures can then be calculated

$$T_i^{n+1} = T_i^n - \frac{(\gamma - 1)\, V_i^{n+1} p_i^n}{N_i} \frac{\Delta t}{\Delta r} \left(u_{i+1}^{n+1} - u_i^{n+1}\right)$$

5. Using the updated temperatures, new thermodynamic pressures can then be calculated using the ideal gas equation of state (p in the following equation is just the thermodynamic pressure)

$$p_i^{n+1} = \frac{N_i}{V_i^{n+1}} k T_i^{n+1}$$

6. Finally, the new time step size is calculated

$$\Delta t = min\left(\frac{\Delta r_i}{c_{s,i} + |u_i^{n+1}|}\right)$$

It is important to note that when ionization is turned on, the number of particles in each zone may change due to the dissociation of electrons and atomic nuclei. When ionization is considered, another step must be added after the density calculation to recalculate the number of particles in each zone.

When thermal conduction, viscosity, and mass diffusion are turned on, the mass fraction equation is solved between steps 4 and 5. Also the velocity, temperature and mass fraction equations are solved implicitly. Aside from these differences, however, the equations are solved in the same order.

4 Test Problems and Verification

Before beginning any real ICF simulations, we conducted several tests to ensure the proper functioning of the individual components of our codes. These tests provided valuable insight into the functioning of the code, and allowed us to move confidently to the ICF simulation phase. The results of the tests are presented here.

4.1 Sod Shock Tube

The well-known Sod Shock Tube problem provides a valuable tool for code verification, given that its exact solution is readily available. We constructed a simulation on the interval $[0, 1]$ with a temperature and pressure discontinuity at 0.5. The system was allowed to evolve until $t = 0.2$ s, and the quantities of interest were compared with the exact solution. All diffusion was disabled for this test.

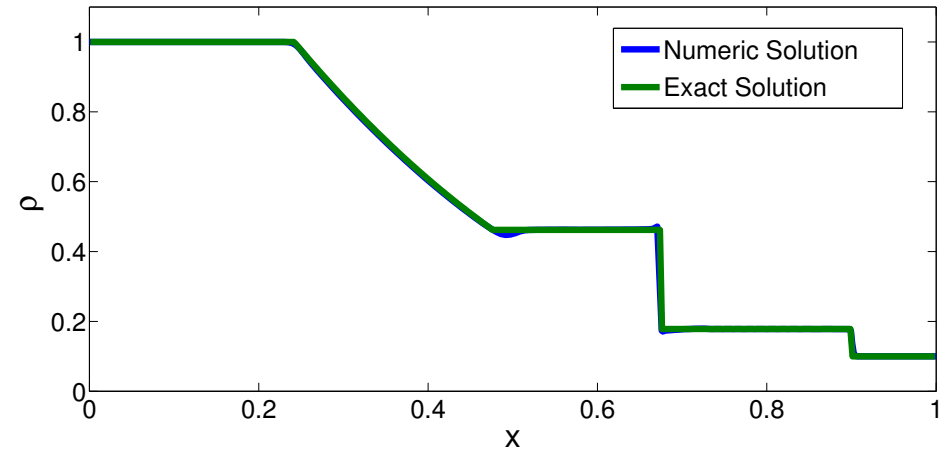
We used the Fortran code provided by Toro [5] to generate data for the exact solution. These solution points were regularly-spaced, as opposed to the solution points from the hydrocode, which were irregularly spaced. We used a matlab script to interpolate the points from the numerical solution and map it onto the exact solution.

Fig. 1a contains both the exact and computed solutions for the density. The agreement is very good, with noticeable artifacts at the shock front and the tail of the rarefaction wave, as well as the contact discontiuity. From Fig. 1b, which plots the relative error in the computed solution, we see that error exists almost exclusively in these three regions. The average l_2 error norm over the entire domain was 0.12%, indicating that the code performed very well for this test.

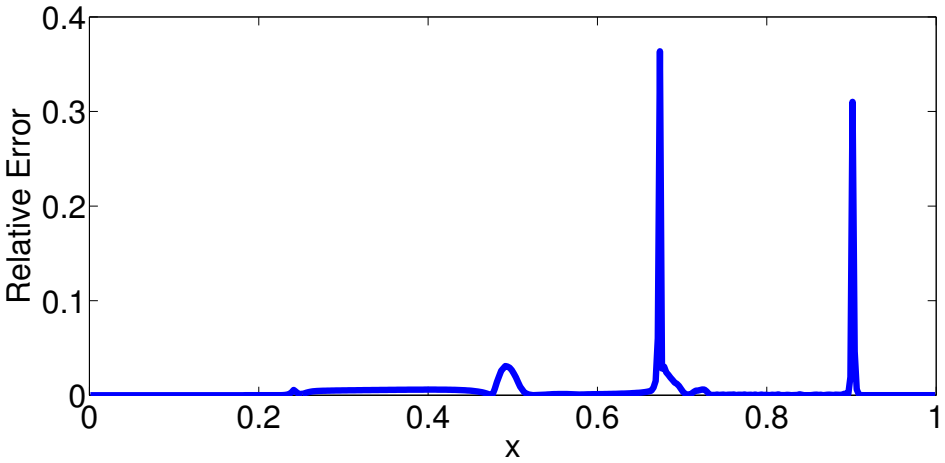
4.2 Surrogate Guderley

The second test problem used to validate the implementation of the pure hydro equations was the Guderley problem. The Guderley test problem has a semi-analytic solution and is a 1D problem with spherical geometry and an inward propogating shock, making it ideal for the validation of the pure hydro equations describing ICF implosions. The test problem is initialized with an excess of energy in the outmost zones in the domain, causing a pressure disparity between the zones with excess energy and those without. The outermost zones were also initialized with a larger density than the interior zones furthercontributing to the difference in pressure. At $t = 0$ the pressure jump causes compression of the inner zones and a shock propogates toward the origin. Throughout the simulation a static outer boundary condition was imposed. It is important to note that the data set used for validation was actually from a Surrogate Guderley problem [6]. This Surrogate Guderley is still an appropriate validation tool as long as the comparison is made when the shock is close to the origin.

Fig. 2a shows a plot of both the semi-analytic and numerical solutions for the pressure profile at 16 seconds (soon after the shock reflects off the origin), and Fig. 2b shows the percent error between these two solutions. The agreement was not as close as with the Sod problem; however differences between the numeric and semi-

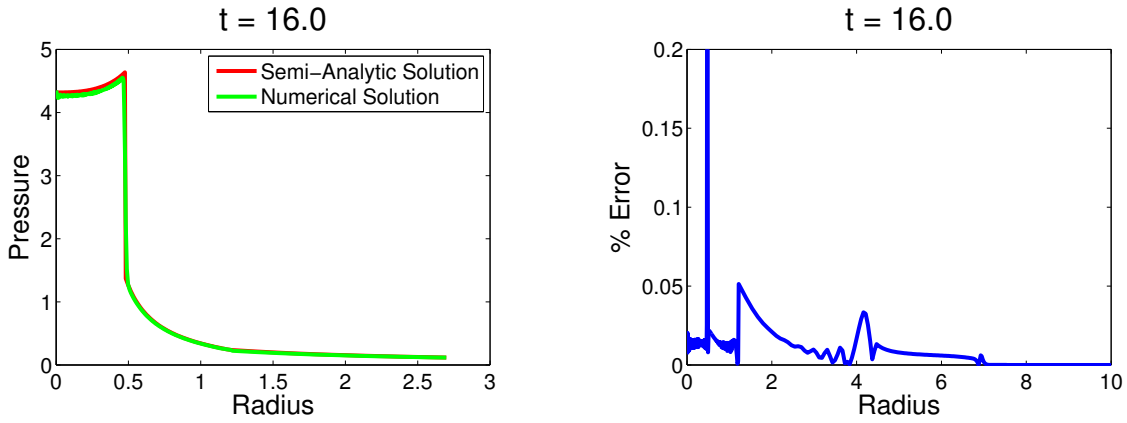


(a) Density profile at $t = 0.2$ s.



(b) Relative error in the computed solution.

Figure 1: Comparison of the computed and exact solutions for the Sod Shock Tube.



(a) A comparison between the numerical solution and the semi-analytics to the surrogate Guderley test problem.

(b) A plot of the percent error between the two solutions. The average error for the entire domain (on the interval [0,10]) was about 1.12%.

Figure 2: The Guderley problem

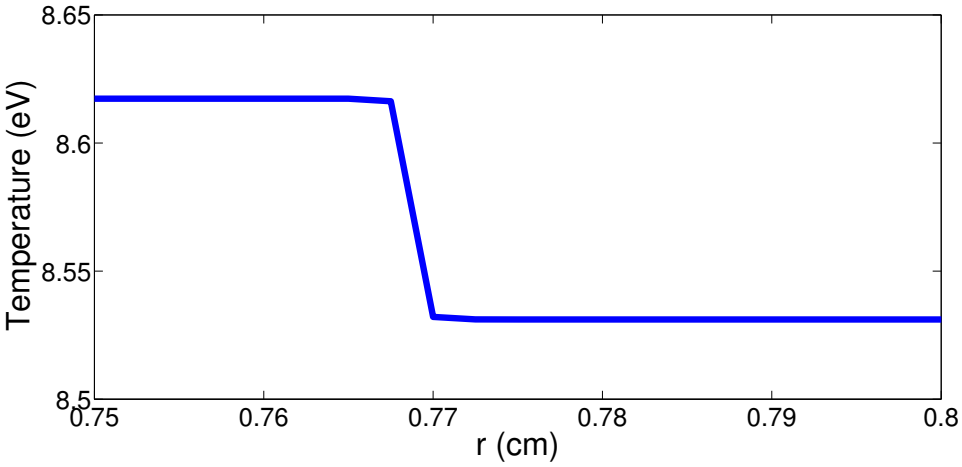
analytic solutions were fairly small. Aside from at the location of the shock, errors were less than 5% throughout the domain (on the interval [0,10]), and the average error between the two solutions was 1.12%.

4.3 Diffusion Test

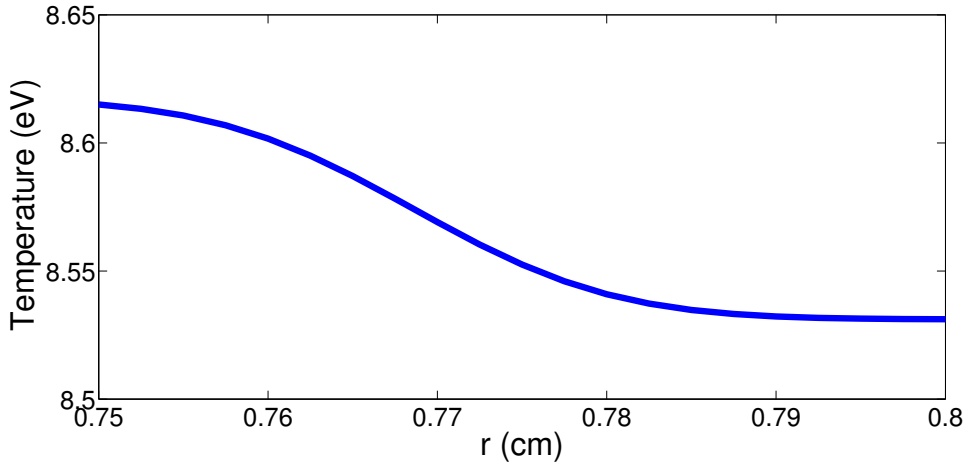
While these tests provided valuable insight into the functioning of the purely hydrodynamic aspects of the codes, they did not address the issue of diffusion. To address this issue, we constructed a simulation of a system in pressure equilibrium, but with a discontinuity in temperature, as illustrated in Fig. 3a. Due to the pressure equilibrium, a system constructed in this way remains static if there is no diffusion, making it an ideal way to test the functioning of the diffusion alone.

After running the simulation, we compared the analytic diffusion length with the diffusion length calculated from the hydrocode's output at each time step. The analytic diffusion length is given by $2\sqrt{Dt}$, where D is the diffusion coefficient. The computed diffusion length is calculated by determining the slope between the two points in the simulation defining the center of the discontinuity. We find the point at which this slope intersects with the lower initial temperature value, and the distance between the center of the discontinuity and this point is the computed diffusion length.

Ideally, this simulation would be run in a planar geometry. However, due to the fact that we implemented the diffusion terms in exclusively spherical geometry, we



(a)



(b)

Figure 3: Initial and final states for the diffusion test.

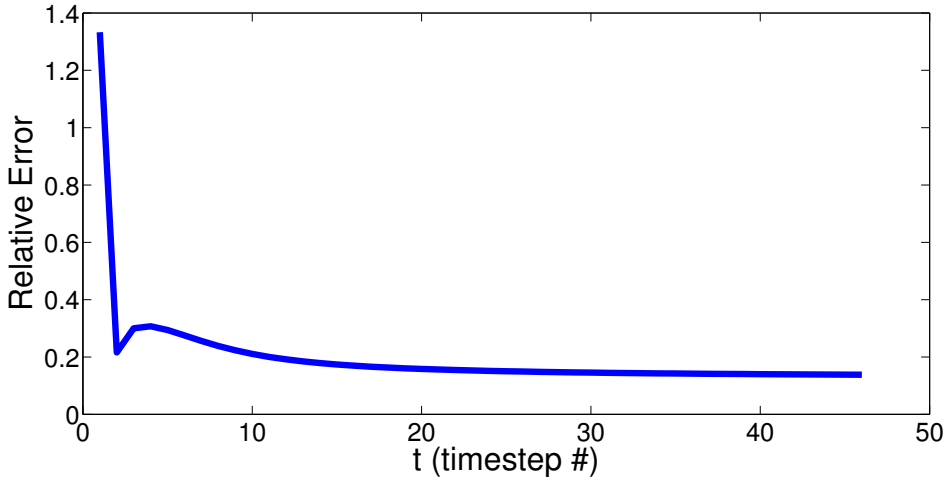


Figure 4: Approximate error in diffusion length

were unable to perform this test in true planar geometry. As an approximation, we constructed the initial conditions such that the discontinuity occurred at a radius that was very large compared to the size of the discontinuity.

Qualitatively, the diffusion performed as expected, with the discontinuity smoothing itself out in a predictable fashion. The temperature state of the system at $t = 6.1 \times 10^{-11}$ s is shown in Fig. 3b. Comparison with the analytic solution gave acceptable results, with the relative error rapidly approaching 13% after the first few time steps. Fig. 4 illustrates this result. Given the uncertainties involved in calculating the actual diffusion length, as well as the error added by the use of a spherical coordinate system, these results were satisfactory.

5 ICF Setup

Following successful testing of the codes, we began the ICF simulation phase. We elected to use a setup similar to that used in ICF simulations conducted using the HELIOS code [7]. HELIOS is a 1D, Lagrangian hydrocode that includes extensive atomic physics, including radiation. Providing inputs similar to those used in HELIOS ICF simulations enabled a coarse level of code verification. Several sources of uncertainty existed in these comparisons; these will be detailed in Sec. 6.

The parameters used corresponded to a typical Omega-scale direct-drive ICF implosion, in which the ICF capsule consists of a deuterium gas surrounded by a plastic

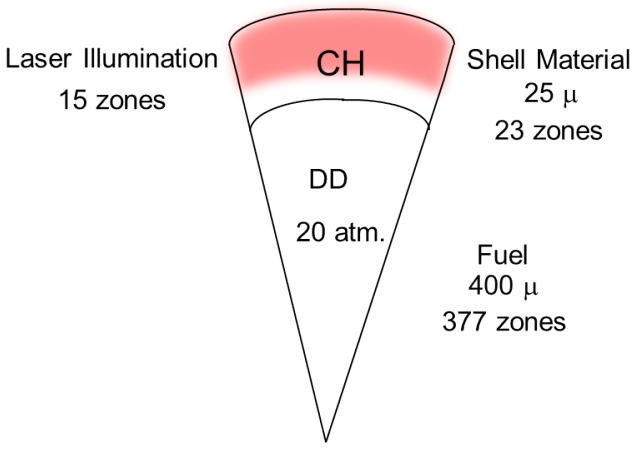


Figure 5: Layout of the simulated ICF capsule.

shell. Both materials were modeled as ideal gases in pressure equilibrium, at a pressure of 20 atm. The fuel was assumed to be at room temperature (300 Kelvin), from which its density was calculated. The shell density was assumed to be one-hundred times that of the fuel.

Fig. 5 illustrates the chosen layout, in which each computational zone corresponds to roughly one micron, and the shell thickness is one-sixteenth that of the fuel radius. The outer fifteen zones represent the laser illumination region. The temperature of the zones in this region was artificially augmented at each time step for the duration of the pulse to represent the delivery of the laser energy. We assumed a square laser pulse with a duration of 1 ns, as shown in Fig. 6, and an absorption efficiency of 30%. The energy was delivered uniformly to all of the zones in the illumination region. At each time step, the temperature of each of these zones was increased by an amount T', given by

$$T' = \frac{2\alpha P_{laser} dt}{3Nd},$$

where P_{laser} represents the laser power of 22 TW, α is the absortion efficiency, N is the total number of particles in the zone, and d represents the penetration depth of the laser energy, in zones.

An important consideration in the ICF setup is the implementation of boundary conditions consistent with those in an actual ICF experiment. We attempted to duplicate such conditions at the outer boundary by fixing the outer guard cell at atmospheric conditions throughout the simulation. This allowed the outer boundary of the plastic shell material to react in a natural way to the applied laser energy. We chose the inner boundary conditions to reflect the spherical symmetry of the system.

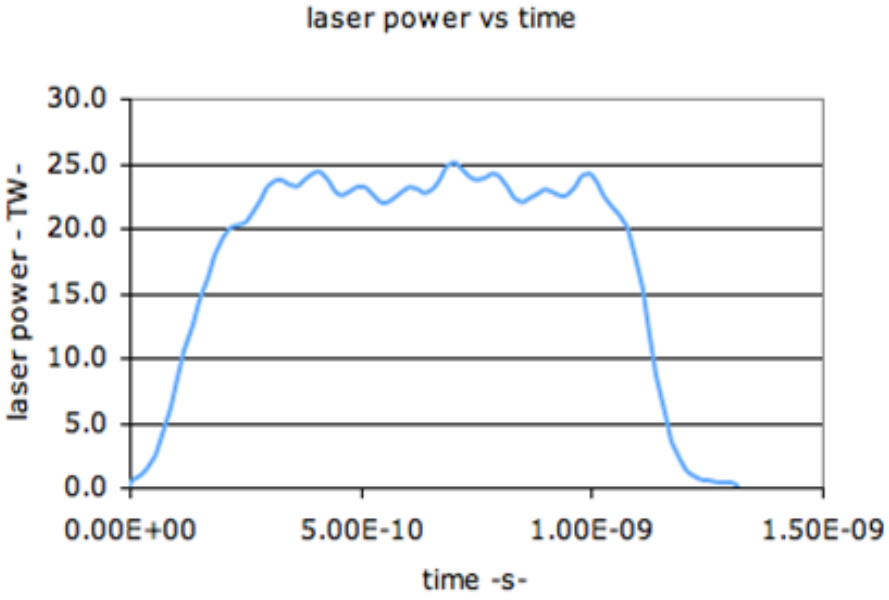


Figure 6: Typical Omega facility laser pulse

As a result, the velocity of the zone boundary at the origin was fixed at zero, and cell-centered quantities in the innermost zone were assumed to be identical to those in a "ghost cell" on the other side of the origin. The dimensions of the ghost cell were also assumed to be identical to those of the innermost zone.

As a final consideration in the ICF setup, we assumed that all post-shock material was completely ionized. Simulations revealed the temperature of the shocked gas to be ~ 100 eV–well above the ionization energy of deuterium–indicating that this was indeed a valid assumption.

6 Results

One of the most telling results in our study came from looking at the peak temperture in the innermost zone as a function of spatial resolution. If we first consider a model where diffusion is turned off, we observe that as the spatial resolution increases, and each of the zones becomes smaller, the peak temperature appears to grow without bound. However, when we then turn on the thermal conduction and plasma viscosity, the peak temperature converges to a finite value as spatial resolution increases. Clearly from Fig. 7, plasma transport effects have a crucial role in controlling the peak

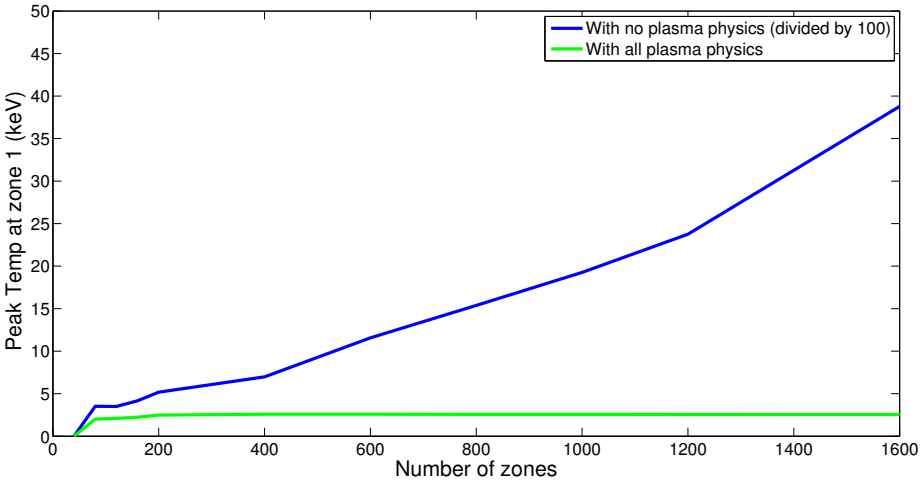


Figure 7: The figure shows peak temperature in the innermost zone versus number of zones (spatial resolution). The blue trace represents simulations that were run with no plasma physics effects activated, and the green trace represents simulations that were run with all plasma physics effects activated. Peak temperatures in the simulations with no plasma physics included are divided by 100 in this graph for aesthetic reasons.

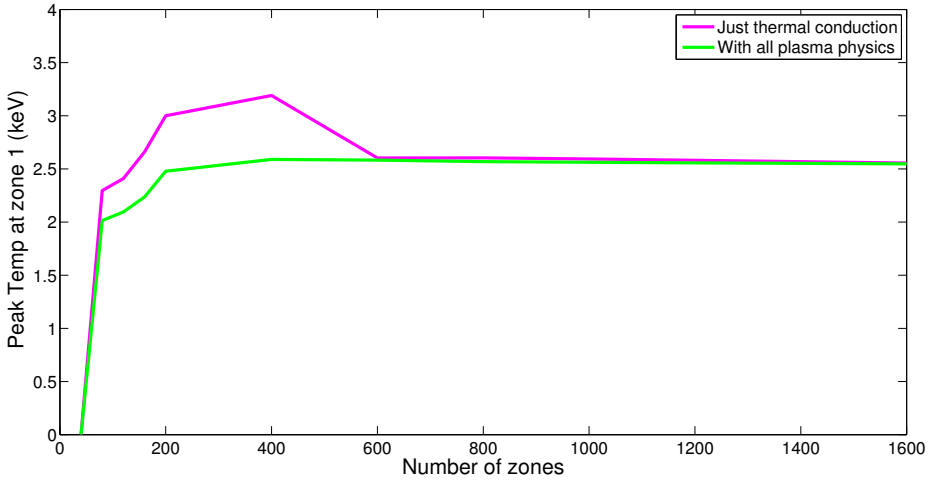
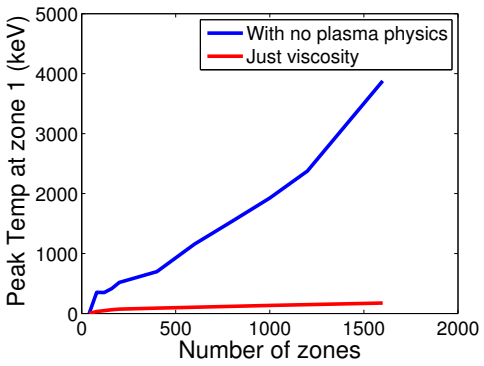


Figure 8: In this plot of peak temperature versus spatial resolution simulations that only include thermal conduction are compared against simulations where all plasma physics effects were included. Thermal conduction apears to be responsible for much of the convergence of peak temperatures, particularly at higher resolutions.

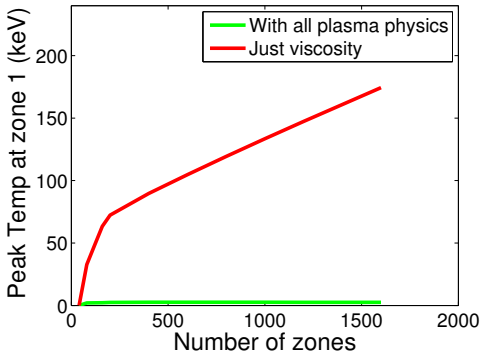
temperatures at the origin. However, we can also develop a qualitative picture for how thermal conduction and viscosity influence peak energies individually by running simulations with only effect or the other activated.

When simulations with only thermal conduction activated are compared to simulations with all plasma effects activated, it is apparent that thermal conduction has a large effect on the convergence of peak temperatures, as shown in Fig. 8. At high resolutions in particular, the thermal conduction simulations almost exactly overlay those with all plasma physics effects active. Viscosity also seems to have an effect on peak temperatures, though not as significant as the effect of thermal conduction, and viscosity likely accounts for the majority of the difference between the two plots in Fig. 8 at low resolution. From Fig. 9 it is clear that viscosity has a significant effect on peak temperatures compared to simulations where no plasma physics effects are included. Fig. 9 also shows, however, that simulations with only viscosity enabled have much greater peak temperatures than when all plasma physics effects are enabled implying that the effect of viscosity is much less significant than the effect of thermal conduction.

Fig. 10 demonstrates the net effect of all the additional diffusion physics, by comparing with pure hydro simulations at various points in time. The smoothing effects of



(a) This is a comparison of simulation which did not include any plasma physics effects with simulations that included only viscosity.



(b) This is a comparison between "viscosity only" simulations and simulations that included all plasma physics. Viscosity has a clear effect on peak temperatures, but the influence of thermal conduction is still most significant.

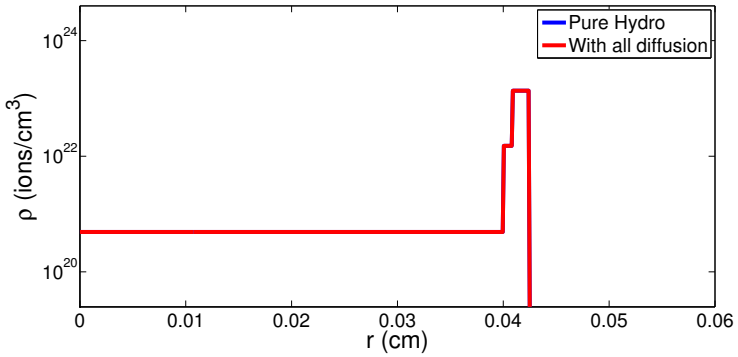
Figure 9: Analysis of plots with only viscosity enabled

the diffusion are easily observed. It is significant to note that in the simulations (run by Daniel) that Fig. 10 are based on, the effect of the added plasma physics causes the shock to speed up compared to simulations with no plasma physical effects. In Ryan's simulations, however, plasma physical effects caused the shock to slow down relative to pure hydro simulations.

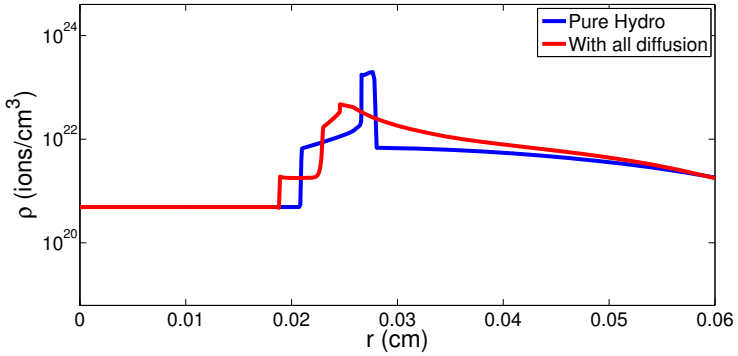
6.1 Comparison to Helios

As detailed in Sec. 5, we specified input parameters similar to those used in ICF simulations carried out using the HELIOS code. In this section, we present a brief comparison of our results with those produced by HELIOS.

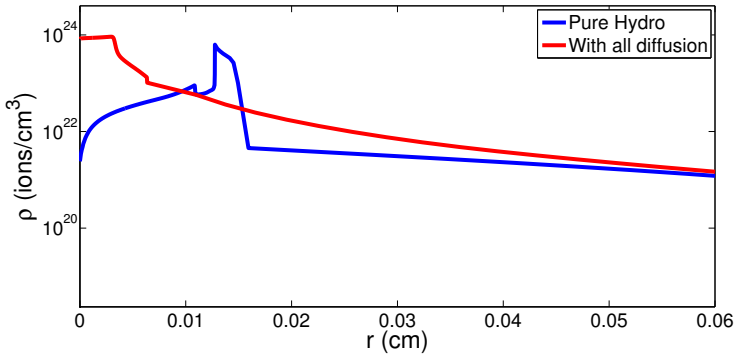
It is important to note when comparing these results that several potentially significant differences exist between the simulations run using the respective codes. The HELIOS simulations take into account radiation, while our codes do not, the implications of which we will attempt to highlight. Another important factor is the fact that the HELIOS simulations do not include viscosity. We will attempt to provide as accurate a comparison as possible by showing two comparisons with the HELIOS results: the first in which viscosity was disabled in our simulations, and a second that includes the effects of viscosity. As a final consideration, we note that we did not know the laser absorption efficiency used in the HELIOS simulations. The value of 30% that we chose seemed a reasonable estimate. Due to these factors, we use the



(a) $t = 0$ s



(b) $t = 5.5 \times 10^{-10}$ s. In both cases, the shock has not yet reached the center.



(c) $t = 9.25 \times 10^{-10}$ s. Reflected shocks are discernable in both cases.

Figure 10: Density profiles for simulations with and without diffusion physics. Note the smoothing effect of the diffusion.

	HELIOS	**Ryan**	**Daniel**
Peak T (keV)	2.8	3.5	10
Peak density (ions/cc)	1.2×10^{25}	2.1×10^{26}	2.6×10^{27}

Table 1: Simulations run without viscosity effects for comparison with HELIOS results.

	Ryan (viscosity included)	**Ryan effect of viscosity**	**Daniel (viscosity included)**	**Daniel effect of viscosity**
Peak T (keV)	3.5	1% increase	4.3	60% decrease
Peak density (ions/cc)	2.2×10^{26}	5.8% increase	4.5×10^{24}	decreased by 3 orders of magnitude

Table 2: Effects of viscosity on peak temperatures and densities.

HELIOS results as a loose guideline only.

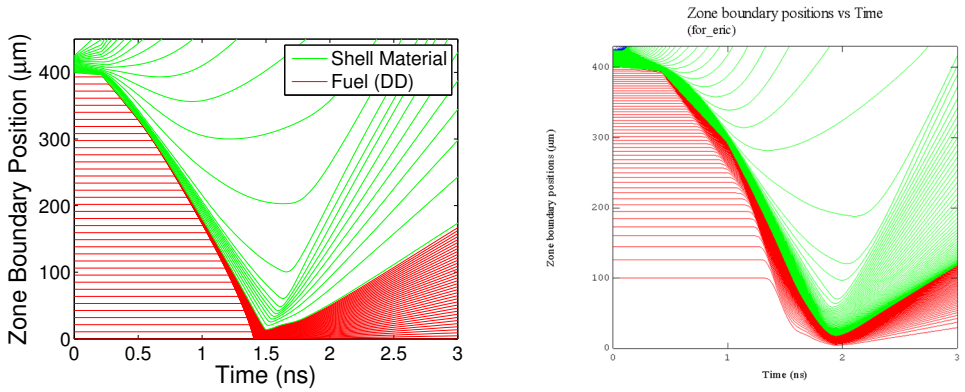
Table 1 gives the comparison to HELIOS results in which viscosity effects were not included in our simulations. Notice that both Ryan and Daniel's codes give peak temperatures and densities that are noticeably higher than those given by HELIOS. There are several likely contributing factors to this discrepancy, the first being the possible difference in laser efficiencies used. Additionally, the HELIOS simulation used very few zones for the gas material, so that the peak density in the innermost zone is expected to be smaller, since the zone itself is larger. Finally, radiation likely plays a role in the HELIOS outcomes–a role that is absent in our results.

Table 2 summarizes the effect of including viscosity in the simulations. Note the modest effect seen in Ryan's code, compared with the dramatic effect exhibited in Daniel's. We postulate that this disagreement between the codes stems at least in part from temperature differences which are magnified by the $T^{5/2}$ term in the diffusion coefficients. We will revisit this idea in more detail in Sec. 6.2. Interestingly, notice that this change has the effect of bringing Daniel's code into much closer agreement with the HELIOS code than previously. Importantly, in both codes, the effects of viscosity seem non-negligible–even for 1D simulations. This is notable, given the fact that viscosity is not widely included in ICF simulations.

A summary of the comparison between HELIOS results and our results with all

	HELIOS	Ryan	Daniel
Peak T (keV)	2.8	3.5	4.5
Peak density (ions/cc)	1.2×10^{25}	2.2×10^{26}	5.7×10^{24}

Table 3: Comparison of simulations including all diffusion physics with HELIOS results.



(a) Our simulation

(b) Helios simulation [7]

Figure 11: Zone boundary positions as a function of time

diffusion terms included is shown in Table 3. The peak temperatures are in good agreement, and the peak densities seem consistent with HELIOS results, given the above-mentioned differences in the problem setups. In Fig. 11 plots of boundary location versus time are displayed for our results and for the Helios code. Qualitatively our results are very similar, and the minimum radius in each case is similar as well. However, the time at which the shell material is blown back out by the shocks is different in the different simulations. Also it is important to note that the spatial resolution in the Helios simulations is coarser near the origin, which may explain the large discrepancy in density between our codes and the Helios code.

6.2 Code Comparison

The motivation for the independent development of two hydrocodes arose from the ability of this arrangement to facilitate code verification. The results presented in Sec. 6.1 highlight the utility of such an arrangement, as evidenced by the discrepancies in results furnished by the codes. It is important to note that at the time of this writing, the verification process is ongoing. We have identified possible sources of

discrepancy between the codes, and, given adequate time, are confident that these differences could be resolved.

As mentioned in Sec. 6.1, we have observed temperature variations between the codes that become magnified as the shock converges toward the center of the sphere. With all diffusion disabled, agreement between the two codes is much better. It seems likely that the diffusion coefficients, all of which depend on $T^{5/2}$, are strongly augmenting the initial differences in temperature. We have further observed that the peak densities and temperatures seem relatively sensitive to differences in problem setup–for instance, the penetration depth of the laser energy and laser absorption efficiency. The combined effects of temperature-difference magnification by means of the diffusion coefficients and relative sensitivity to early conditions likely account for the majority of the inconsistencies we observe.

An interesting next step would be to implement radiation physics. The higher temperatures seen in Daniel's code would result in greater energy loss by radiation, likely bringing the results in greater alignment with Ryan's, though the extent of this effect is unknown. It is possible that the differences seen are merely the result of modeling the same physical setup in two different ways, and that the inclusion of more realistic physics, such as radiation, would allow both codes to converge toward a more physically-realistic solution.

Despite these differences, the fact remains that both codes gave peak energies similar to the HELIOS simulation results, and peak densities that seem consistent with these results.

7 Conclusions

The hydrocodes we have developed are still a work in progress; this notwithstanding, some important conclusions can be drawn from current results. We have seen that thermal conduction plays a crucial role in ICF, and is essential for accurate simulations. The importance of viscosity in 1D ICF simulations is less pronounced. It appears from our simulations that the effect may be important, but further study is necessary to determine the magnitude of this effect. The viscosity is more likely to play a role in high resolution 2D or 3D simulations including drive asymmetries. The species diffusion proved to be an area of significant disagreement between our codes, with Ryan's results showing very little mixing, and Daniel's indicating a large degree of mixing at the boundary layer. These results may reflect the sensitivity of the diffusion coefficients to differences in temperature. Result differences may also be related to the code difference in the diffusion gradient drive where the mass fraction was used in one code (Ryan's) and the molar fraction was used in one case (Daniel's).

The differences are inconclusive and will require further study.

Additionally, we conclude from this work that comparison between two independently-developed codes is a valuable tool for verification. The differences between our two codes highlight the fact that coding is a difficult process, and the results should be treated with a degree of skepticism. We see as well that the effects of differing approaches to the same problem can be significant. It is our hope that as the codes become more physically realistic by the inclusion of more sophisticated physics, these differences will be increasingly reduced.

8 Future Work

There are several natural extensions to the work presented here, which can and should be pursued in future investigations of ICF modelling. One improvement to our codes that could be implemented immediately is to add radiation transport terms to the temperature equation. It is unclear how great an effect this physics would have on our results; however the model implemented in the Helios code includes the effects of radiation transport, so incorporating it into our codes would provide a better basis for comparison. A logical next step would be to adapt our codes to 2D geometry. A 2D spherical geometry code would allow for the application of asymmetrical laser forcing in the outer shell of the simulated capsule, and it would allow us to see the effects of this asymmetry on peak temperatures and densities at the center of the capsule. Viscosity was seen to have at least a modest influence on the the peak values in the 1D case; however as fluid instabilities and turbulence arise from asymmetrically applied laser forcing [8], then viscous effects could become quite significant. After building a 2D Lagrangian code, the next step would be to remap the simulation to the Eulerian frame so that the instabilities that arise from the asymmetrical forcing can be analyzed in greater detail without being subject to Lagrange mesh tangling.

References

[1] Atzeni, S., 'The physical basis for numerical fluid simulations in laser fusion', Plasma Phys. Control. Fusion 29 1535 (1987).

[2] Wilson, D.C., Ebey,P.S., Sangster,T.C., Shmayda,W.T., Glebov,V.Yu, etal., 'Atomic mix in directly driven inertial confinement implosions', Phys. Plasmas, 18, 112707 (2011).

[3] Braginskii, S.I., 'Transport Processes in a Plasma', Reviews of Plasma Physics, Vol. 1 (Consultant Bureau, New York, 1965), p. 205

[4] Pember, R.B., Anderson, R.W., 'A Comparison of Staggered-Mesh Lagrange Plus Remap and Cell-Centered Direct Eurlerian Godunov Schemes for Eulerian Shock Hydrodynamics', Nuclear Explosives Code Developers Collaborations (NECDC) 2000 Oakland, CA October 23-27, 2000, Lawrence Livermore National Laboratory (2000) .

[5] Toro, E., Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer-Verlag Berlin Heidelberg 2009.

[6] Ramsey, S., Shashkov, M.J., 'Surrogate Guderley Test Problem Definition', Los Alamos National Laboratory unclassified report LA-UR-12-22751, Los Alamos, NM (2012).

[7] MacFarlane, J.J., Golovkin, I.E. Woodruff, P.R., 'HELIOS-CR A 1-D radiation-magnetohydrodynamics code with in-line atomic kinetics modeling', J.Quant. Spectro. and Rad. Transfer, 99 381-397 (2006).

[8] Thomas, V.A., Kares, R. 'Drive asymmetry and the origin of turbulence in ICF implosions', Los Alamos National Laboratory unclassified report LA-UR-11-03527, Los Alamos, NM (2011).

Acknowledgements

We would like to acknowledge the Computational Physics Student Summer Workshop for providing us with the opportunity to conduct this research. We would, in particular, like to thank Scott Runnels for all the hard work he put forth to make the workshop a success. We would also like to thank Scott Ramsey for providing us with valuable semi-analytic test problem data.

An Improved Time Step Controller for xRage

(Tom Masser, mentor)

An Improved Time Step Size Control for xRAGE's 3-T Plasma Code

Catherine M. Gosmeyer

Partner: Brandon K. Wiggins

Adviser: Thomas Masser

Los Alamos National Lab

Computational Physics Workshop Summer 2013

LA-UR pending

August 15, 2013

Abstract

The time step size algorithm used in xRAGE's 3-T plasma code could be optimized. In order to resolve all of the detail in the equilibrating system, the present algorithm picks a very small initial step size and grows it at an insignificant rate, even when the system has reached equilibrium and could be fully resolved with a larger step size. Here we develop an improved algorithm that grows or shrinks the step size as appropriate to capture all of the system's anomalies, and as result, may speed up the wall time. We also describe how we implemented the algorithm into the xRAGE code.

1 Introduction

The three-temperature (3-T) coupling equations in LANL's xRAGE code may become stiff. Stiffness occurs when the numerical methods being used to solve the equations are unstable, forcing the time step size to be very small in order to obtain satisfactory results (Jackiewicz 2009). A constantly small step size is highly inefficient. In its present state, the xRAGE code uses a slightly *ad hoc* algorithm to vary step sizes; in order to obtain the best results, it is best to always use a small step size. The code can be vastly improved if we can develop an algorithm that can grow and shrink the step size as appropriate for the changes that occur in the electron, ion, and radiation temperatures. Our goal in this project is to develop a "golden mean" control, which does not sacrifice accuracy for speed, and whose runtime is still shorter than present controls'. The plot in Figure 1 motivates this goal.

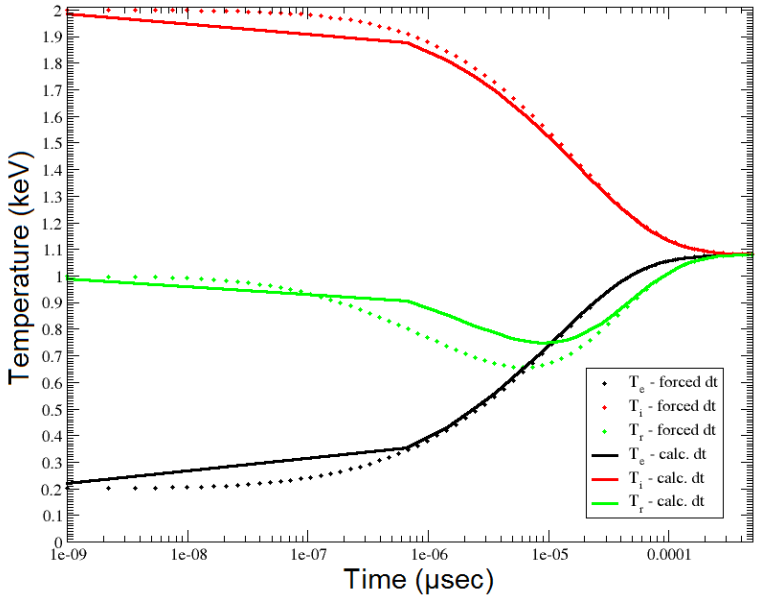


Figure 1: The dotted line shows the throttled integration and the solid line shows a sped-up integration, both using controls presently in xRAGE. There is a noticeable difference between forcing the time step to be small and taking larger step sizes in order to decrease the runtime. An especially poor match is in the radiation temperature, where the sped-up integration misses the true minimum. We hope to develop a smarter control that will take small step sizes only when necessary, without needing to sacrifice accuracy for speed.

Another problem with the 3-T solver is that it arbitrarily assigns the initial step size to a very small value, without taking the physical parameters of the system into account.

xRAGE, the "Radiation Adaptive Grid Eulerian" code, is a cell by cell adaptive mesh refinement Eulerian radiation hydrodynamics code. It contains 3-T plasma physics, which xRAGE solves by operator splitting in time. A more complete description is given in Gittings et al. (2008) and McClarren & Wöhlbier (2010).

xRAGE's plasma model is used in many astrophysical problems, such as supernova simulations. It is of interest, then, to find a smarter and faster algorithm.

2 The 3-T Equations

Plasma models often assume two temperatures, an electron temperature and an ion temperature. If the plasma model is coupled with a radiation diffusion model, in which radiation energy density can be described by a radiation temperature, we obtain a three-temperature (3-T) model.

xRAGE contains such a 3-T plasma physics model, which it solves for one-dimensional spatial problems using these six PDEs:

$$\frac{\delta\rho}{\delta t} + \nabla \cdot \rho\mathbf{u} = 0 \tag{1}$$

$$\frac{\delta}{\delta t}\rho\mathbf{u} + \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u} + P_e + P_i) + \frac{1}{3}\nabla E_r = 0 \tag{2}$$

$$\frac{\delta\rho e_i}{\delta t} + \nabla\cdot(\rho e_i\mathbf{u}) + \nabla\cdot\mathbf{q}_i + P_i : \nabla\mathbf{u} = \gamma_{ei}(T_e - T_i) + \dot{S}_i \tag{3}$$

$$\frac{\delta\rho e_e}{\delta t} + \nabla\cdot(\rho e_e\mathbf{u}) + \nabla\cdot\mathbf{q}_e + P_e : \nabla\mathbf{u} = \gamma_{ei}(T_i - T_e) + c\sigma_a(E_r - aT_e^4) + \dot{S}_e \tag{4}$$

$$\frac{\delta}{\delta t}\rho E + \nabla\cdot[(\rho E \mid +P)\cdot\mathbf{u}] + \nabla\cdot(\mathbf{q}_e + \mathbf{q}_i) = c\sigma_a(E_r - aT_e^4) - \frac{1}{3}\mathbf{u}\cdot\nabla E_r + \dot{S}_i + \dot{S}_e \tag{5}$$

$$\frac{\delta E_r}{\delta t} + \frac{4}{3}\nabla\cdot(\mathbf{u}E_r) - \nabla\cdot(\kappa\nabla E_r) = -c\sigma_a(E_r - aT_e^4) + \frac{1}{3}\mathbf{u}\cdot\nabla E_r. \tag{6}$$

See Wöhlbier (2007) for a more in-depth description of the PDEs.

If we remove spatial dependence, we obtain three ODEs. These are simpler to work with and therefore we use them as a starting point for developing our own solvers to test our new step size controls.

The 3-T ODE coupling equations to be solved are

$$\frac{\delta E_r}{\delta t} = c\sigma_a(aT_e^4 - E_r) \tag{7}$$

$$\frac{\delta\rho e_e}{\delta t} = \gamma_{ei}(T_i - T_e) + c\sigma_a(E_r - aT_e^4) \tag{8}$$

$$\frac{\delta\rho e_i}{\delta t} = \gamma_{ei}(T_e - T_i) \tag{9}$$

We define radiation temperature as $E_r = aT_r^4$ and rescale the radiation interaction coefficient

$$\hat{\sigma} = \sigma_a\frac{T_e^4 - T_r^4}{4T_r^3(T_e - T_r)}.$$

Letting $\frac{1}{2\tau_R} = c\hat{\sigma}$ and $\frac{1}{2\tau_P} = \frac{\gamma_{ei}}{\rho C_{v,i}}$, the final reduced form is

$$\frac{\delta T_r}{\delta t} = \frac{T_e - T_r}{2\tau_R} \tag{10}$$

$$\frac{\delta T_e}{\delta t} = f_1\frac{T_r - T_e}{2\tau_R} + f_2\frac{T_i - T_e}{2\tau_P} \tag{11}$$

$$\frac{\delta T_i}{\delta t} = \frac{T_e - T_i}{2\tau_P}. \tag{12}$$

Two of the four parameters of the system are

$$f_1 = \frac{C_{v,i}}{C_{v,e}}$$

and

$$f_1 = \frac{4aT_r^3}{\rho C_{v,e}}.$$

The two other parameters are the radiation and plasma relaxation times, τ_R and τ_P. The temperature of the radiation, electron, and ion components are, respectively, T_r, T_e, and T_i. The density is ρ, the radiation constant is a, and the specific heats for electrons and ions are $C_{v,e}$ and $C_{v,i}$.

We introduce here a backward Euler (implicit) method for solving the 3-T coupling equations. The implicit equations are obtained by first converting the above 3-T equations into matrix form,

$$\frac{\delta}{\delta t}\mathbf{x} = \mathbf{A}\mathbf{x},$$

in which,

$$\mathbf{A} = \begin{pmatrix} -\frac{1}{2\tau_R} & \frac{1}{2\tau_R} & 0 \\ \frac{f_2}{2\tau_R} & -\left(\frac{f_2}{2\tau_R} + \frac{f_1}{2\tau_P}\right) & \frac{f_1}{2\tau_P} \\ 0 & \frac{1}{2\tau_P} & -\frac{1}{2\tau_P} \end{pmatrix},$$

with the eigenvalues

$$\lambda = 0, -\frac{\tau_P + f_2\tau_p + \tau_R + f_1\tau_R \pm \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4\tau_P\tau_R(1 + f_1 + f_2}}{4\tau_P\tau_R}$$

We invert the matrix and solve for the updated temperatures,

$$T_r^{n+1} = \frac{(1 + 2R + P + RP)T_i^n + (P + RP)T_e^n + RPT_r^n}{1 + 2R + 2P + 3PR} \tag{13}$$

$$T_i^{n+1} = \frac{(1 + 2P + R + RP)T_r^n + (P + RP)T_e^n + RPT_i^n}{1 + 2R + 2P + 3PR} \tag{14}$$

$$T_e^{n+1} = f_1(T_i^n - T_i^{n+1}) + f_2(T_r^n - T_r^{n+1}) + T_e^n, \tag{15}$$

where $R = \frac{\Delta t}{2\tau_R}$ and $P = \frac{\Delta t}{2\tau_P}$.

Note that the 3-T equations conserve total energy. In order to obtain the updated electron temperature (Equation 15), we note that

$$\frac{d}{dt}(f_1T_r + T_e + f_2T_i) = 0.$$

2.1 Operator Splitting

The 3-T equations can be split into radiation and plasma components and solved one at a time instead of all at once. There are many splitting methods available in the literature, for instance, Lie splitting and Strang splitting. Here is an example of how to split the equations.

First solve:

$$\frac{\delta \rho e_e}{\delta t} = \gamma_{ei}(T_i - T_e)$$

$$\frac{\delta \rho e_e}{\delta t} = \gamma_{ei}(T_e - T_i)$$

Then solve:

$$\frac{\delta E_r}{\delta t} = c\sigma(aT_e^4 - E_r)$$

$$\frac{\delta \rho e_e}{\delta t} = c\sigma(E_r - aT_e^4).$$

We looked briefly into the difference in the solutions caused by solving the equations by splitting and by solving the equations whole. We did not pursue this analyses far, however, and instead spent most our time investigating step size controls.

2.2 The Test Code

We wrote two solvers for Equations 13, 14, and 15 in order to test our new time step size controls. One was written in IDL and produced plots for comparing step size controls over various permutations of the initial temperatures and the relaxation times. An example plot is shown in Figure 2. These test solvers were necessary because we did not wish to test our controls in xRAGE itself. They allowed us the flexibility to switch in and out step size controls and develop the controls in a simple environment.

In order to incorporate our final controls into xRAGE we needed to translate our C and IDL into Fortran90. We describe this process further in Section 4.

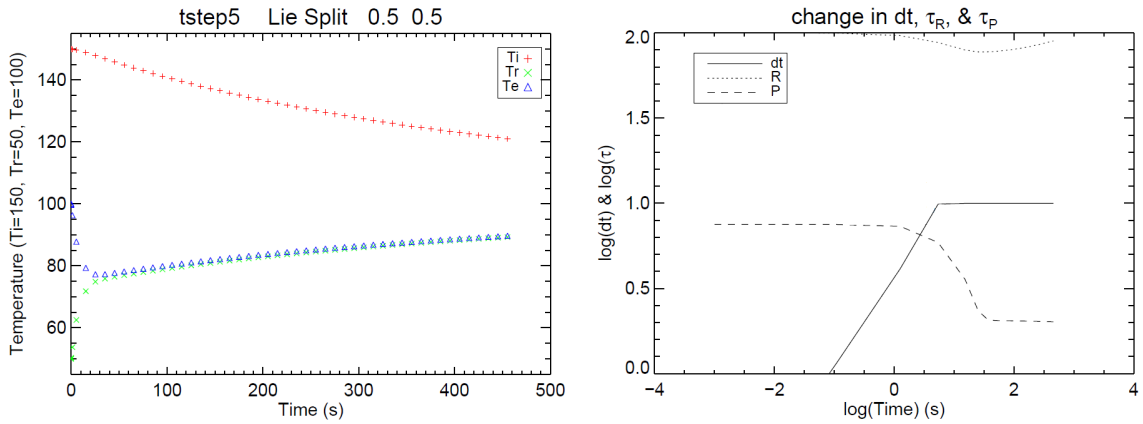


Figure 2: An example of the output of the IDL test solver. The left plot shows the relaxation of the three temperatures using Lie splitting Implicit Euler when the initial relaxation times both are 0.5 seconds and the initial temperatures are $T_i = 150$, $T_e = 50$, and $T_r = 100$ eV. The right plot shows the change in step size and relaxation times with time on a log scale.

3 Present Time Step Size Algorithms

xRAGE currently contains several step size controls for the radiation and plasma components.

3.1 Initial Step Size

For relaxation problems like these, xRAGE currently uses only two parameters, based on plasma and radiation, to choose the initial time step size.

The plasma initial step size is set in the subroutine "threet_inital_tstep" located in the threet.f90 module. There, "3TForce" suggests $\Delta t_{0,P} = 10^{-14}$ seconds. The purpose is to ensure the first step size is small enough so that no rapid changes in the temperatures are overlooked. This algorithm is, however, not based on the physics of the system.

The radiation initial step size is set in the subroutine "inital_tstep" in module_initial.f90. In this subroutine, "RadZero" suggests $\Delta t_{0,R} = \Delta x/5c_{light}$ seconds. This is based on radiation diffusion, but still might be based on more relevant parameters.

The time step controller in module_tstep.f90 receives all of these suggested step sizes, takes the minimum, and divides by 10. The result is the initial step size for the entire system. This initial value ($\Delta t_0 \leq 10^{-15}$ seconds) works well to capture the temperature relaxation in most cases; however, it needs not be as small as it is, and it has the unintended consequence of keeping the subsequent step sizes always very small.

3.2 Subsequent Step Sizes

xRAGE has several options for step size control of subsequent steps. We list four in Table 1 and go into more detail below.

Table 1: Step size control options.
The status lists whether the control is turned on or off by default.

Control	Status	Location
dtev±chg	on	module_radiation_library.f90
dt_splits	off	module_radiation_diff.f90
ei-couple	off	threet.f90
20 pct	on	module_tstep.f90

3.2.1 dtev±chg

The basis of dtev±chg is the change to the final equilibrium temperature from cycle to cycle. A possible problem with this algorithm is that the current step size is based on conditions potentially in the distant future. For uniform domain problems, the equation for dtev±chg's updated step size looks like

$$\Delta t_{new} = \Delta t_{old} max\left(floor, \frac{\eta_{target}}{\eta}\right),$$

in which,

$$\eta = \frac{|T'_{eq} - T_{eq}|}{max(T'_{eq}, T_{eq}) + \epsilon}$$

and

$$\eta_{target} = 0.2 \qquad \text{(de\_tevpct)}.$$

The equilibrium temperature before the radiation update is T_{eq} and the equilibrium temperature after the radiation update is T'_{eq}. The offset to prevent division by zero is ϵ.

3.2.2 dt_splits

dt_splits is based on electron temperature changes from the radiation solver. It does not, however, have a floor. Without a floor the step size can go to zero and the code then grind to a halt. This control is turned off by default.

The equation for the updated time is

$$\Delta t_{new} = \Delta t_{old} \frac{\eta_{target}}{\eta},$$

where

$$\eta = \frac{T_e' - T_e}{max(\epsilon, T_e', T_e)}$$

and

$$\eta_{target} = 0.2 \qquad \text{(siepct)}.$$

The ϵ acts as an offset to prevent division by zero.

Note that the absolute value of the temperature difference is not taken. This means that in the equation for Δt_{new} only positive η's are considered, and therefore $T_e' > T_e$, and so this control only influences Δt when the electron temperature increases. This could be a bug (or an intended feature). We view this effect as a defect.

3.2.3 ei-couple

The equation for the step size control related to electron-ion coupling is

$$\Delta t_{new} = \frac{\rho}{\gamma_{ei}} \frac{min(|e_e|, |e_i|)}{|T_i - T_e|} \eta_{target},$$

where

$$\eta_{target} = 0.4 \qquad \text{(siepct\_3t)}.$$

Similar to dt_splits, ei-couple is based on $|T_i - T_e|$ but also contains no floor. This control too is turned off by default.

3.2.4 20 pct

Finally, 20 pct increases each step size by 20% of the previous step. The percent cycle-to-cycle increase may be set by the user using the code parameter dtfac_grow (default value 1.2).

This has its merits in that the step size is smallest at the beginning of the run, where there are more bumps in the temperature, and it increases as the system goes to equilibrium, where there are fewer areas of interest. This is similar to the control we want to develop; but 20 pct could be more elegant if it were based on physical parameters. And because the initial step size selects such a small number, 20 pct is not as effective as it could be, and the step size remains forever small.

4 The Improved Step Size Algorithm

An improved step size algorithm would be based on relevant physical parameters. Our algorithm makes use of the plasma and radiation relaxation times, here labeled as τ_P and τ_R, which xRAGE already calculates.

4.1 Derivation

Our algorithm has the form,

$$\Delta t_{new} = max\left(\lambda_R \tau_R, \Delta t_{old}\left(\frac{\eta_{target,R}}{\eta}\right)^{1/2}\right)$$

located in module_radiation_diff.f90, and,

$$\Delta t_{new} = max\left(\lambda_P \tau_P, \Delta t_{old}\left(\frac{\eta_{target,P}}{\eta}\right)^{1/2}\right),$$

located in threet.f90.

Here η is a relative temperature change and η_{target} is the target change. If the change overshoots the target, the step size will shrink. Likewise, if the change is smaller than the target, the step size will be allowed to grow.

The floor terms are $\lambda\tau$, where λ is a fraction and τ is the relevant relaxation time.

We gave our control a floor based on a fraction of the relevant relaxation time such that the step size could not become arbitrarily small.

We combined the floor with a well-known time step size control from the literature (see Rider & Knoll (1999)), shown as the second terms in the maximum function above.

4.2 Usage in xRAGE

To turn on our step size control, set

improved_tsc = .true.

Note the following variable names in xRAGE.

- λ_P = ei_dtfac, the fraction of plasma relaxation time to use as a floor for the plasma step size control.
- λ_R = rad_dtfac, the fraction of radiation relaxation time to use as a floor for the radiation step size control.
- $\eta_{P,target}$ = tiepct

  and

- $\eta_{R,target}$ = trepct, the thresholds for relative temperature change used to determine whether the step size should increase or decrease.

4.3 Comparison to Old Controls

We test our new controls against the old in 12 permutations of the initial conditions for three temperature relaxation problems. The results are shown in Table 2. We see by comparing wall times that our algorithm is, in fact, faster. In Figure 3 we see that not only is our algorithm faster, it also still captures the relaxation of the temperatures well.

Table 2: Speed of our new algorithm versus the old for a 10×10 grid. Notice that the new control consistently gives fewer steps and a faster wall time.

	OLD		NEW	
	Num Steps	Wall Time [s]	Num Steps	Wall Time [s]
3t_relax_01	64	36	29	17
3t_relax_02	64	29	33	16
3t_relax_03	64	27	26	27
3t_relax_04	64	26	26	17
3t_relax_05	64	36	33	16
3t_relax_06	64	22	26	17
3t_relax_07	64	22	27	11
3t_relax_08	64	22	32	16
3t_relax_09	64	22	32	13
3t_relax_10	64	21	27	12
3t_relax_11	64	21	27	13
3t_relax_12	64	22	32	13
3t_eir_relax	64	21	64	25
3t_ei_relax	64	8	64	8

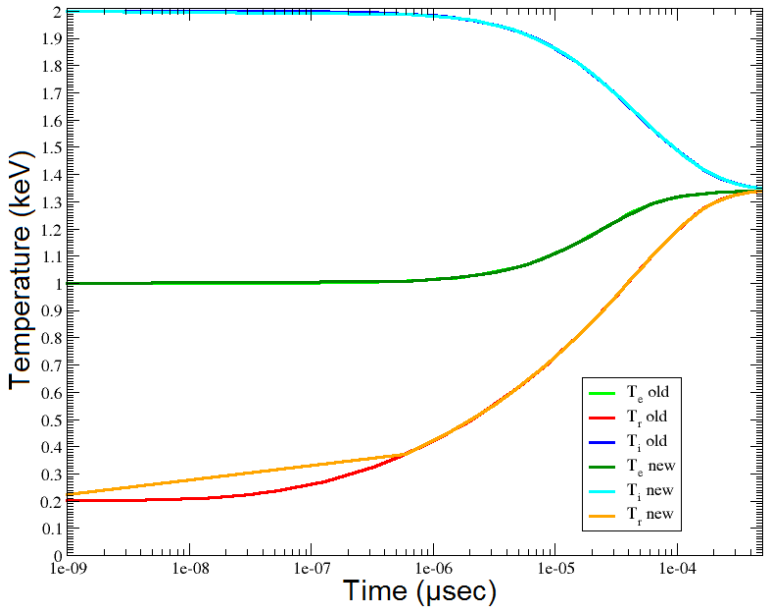


Figure 3: The code is run first with throttled step sizes. Then we run the code with our algorithm and plot over the throttled temperatures. The time is on a log scale. We see a very good match between the new control and the old. The apparent mismatch in the radiation temperature is merely a plotting artifact. Our control (orange) took a long step and hit the correct solution. Only if one needed to see the initial temperature changes in great detail would our control be inadequate.

5 Conclusions

We analyzed the 3-T coupling ODEs and solved them in test codes we wrote to investigate alternative step size controls for xRAGE's plasma physics model. We looked at xRAGE's existing initial and subsequent step size controls, dtev±chg, dt_splits, ei-couple, and 20 pct, and discussed their pros and cons. We saw that many lacked floors or based their step size increments on non-physical parameters. Finally we developed our new control and implemented it in xRAGE. Our control bases the step sizes on plasma and radiation relaxation times and it contains a floor. We found that our controls decrease the runtime without sacrificing too much accuracy in resolving the temperature relaxations.

6 References

Gittings, M., et al. 2008, Comput. Sci. Disc., 1

Jackiewicz, Z. 2009, General Linear Methods for Ordinary Differential Equations, John Wiley & Sons, 38

McClarren, R. G. & Wöhlbier, J. G. 2011, JQSRT, 112, 119-130

Rider, W. J. & Knoll, D. A. 1999, JCPH, 152, 790-795

Wöhlbier, J. G. 2007, LANL Research Note, LA-UR-07-4820

7 Acknowledgments

Thanks to

ASC Hydro Project for funding.
John Wöhlbier for guidance during the first week.
Scott Runnels for organizing and managing the workshop.
Tina Jenkins for technical support.

Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

A NEW TIME STEPSIZE SELECTION SCHEME FOR LOS ALAMOS NATIONAL LABORATORY'S RADIATION HYDRODYNAMICS CODE RAGE

BRANDON K. WIGGINS, KATIE GOSMEYER, AND THOMAS MASSER

Abstract. RAGE's time stepsize selection scheme is discussed and critiqued. RAGE currently employs very small time steps in its integration which may not be required for all initial conditions or simulations. In this paper, we propose an improved algorithm based on three temperature physics and a well-known time step selection criteria. We offer discussion on additional timestep selection schemes which may have merit. We briefly treat the strengths of various operator split methods.

1. Introduction

Los Alamos National Laboratory's code RAGE (Radiation Adaptive Grid Eulerian) is a massively parallel, multidimensional, multi-material, multi-physics code employed by the laboratory in studies of high energy systems. The code solves the Euler equations coupled with radiation diffusion equations to predict a given system's evolution.

RAGE's long developmental history has facilitated its somewhat cumbersome structure. Over the course of the code's development, a variety of physics packages have been added in the aim to improve the fidelity of its simulations as well as expand the set of systems for which RAGE would give good results. In spite of continued work on the code to improve algorithms and integration techniques, RAGE's time stepsize selection scheme (hereafter stepsize selection scheme) remains inefficient and the implementation of a better scheme might have significant benefits. Though the code's structure does not lend itself well to a serious overhaul in this regard, we motivate and document a new timestep algorithm which may be employed in three temperature plasma simulations to a more efficient effect. We will also briefly discuss the stability and other merits of various operator split methods in connection with the three temperature coupling equations.

1.1. The Current Scheme.

RAGE integrates its entire computational domain with a single timestep. In the process of determining its next stepsize, the code gathers recommendations from all of its cells and from the collection of its physics packages. Once the minimum stepsize is determined, the entire system is integrated at this time stepsize.

A weakness in the current time stepping scheme is obvious in its default three temperature physics initial stepsize recommendation. By default, the code employs a "3tFORCE" option which constrains the initial stepsize to 10^{-14} seconds regardless of the nature of the initial conditions. The code's "rad zero" option recommends an initial timesize of $\Delta x_{min}/(5c)$ where Δx is the dimension of the smallest cell and c is the speed of light. After the code gathers recommendations

for the size of the first timestep, it then divides the smallest recommendation by 10 for no apparent reason other than for "good measure."

Inefficiencies further arise in stepsize controls for subsequent timesteps. The "20pct" option guarantees that the stepsize for a subsequent cycle will not get larger than 1.20 times the stepsize of the previous timestep. This stipulation, combined with very small initial stepsizes frequently result in simulations where the code's stepsize is constrained entirely by seemingly *ad hoc* criteria instead of relying on underlying physics to recommend efficient stepsizes. It could be mentioned here that the code's only three temperature plasma time stepsize selection scheme based upon three temperature physics "3t_tsc" is, by default, turned off. An additional stepsize recommendation based on the change in electron temperatures due to the radiation coupling ("dt_splits") is also not employed.

The authors emphasize that the time stepsize selection scheme described above *works*, but the implementation lacks efficiency and elegance. During the course of this paper, we will introduce a new timestep selection scheme which may be used in a variety of applications.

2. Time Stepsize Selection Schemes

The complete set of partial differential equations which RAGE seeks to solve which model electron-ion hydrodynamics and radiation diffusion for a three temperature plasma is

$$\begin{aligned}
\frac{\partial}{\partial t}\rho + \nabla\cdot\rho\mathbf{u} &= 0\\
\frac{\partial}{\partial t}\rho\mathbf{u} + \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u} + \mathrm{P}_e + \mathrm{P}_i) + \frac{1}{3}\nabla E_r &= 0\\
\frac{\partial}{\partial t}\rho e_i + \nabla\cdot(\rho e_i\mathbf{u}) + \nabla\cdot\mathbf{q}_i + \mathrm{P}_i : \nabla\mathbf{u} &= \gamma_{ei}(T_e - T_i) + \dot{S}_i\\
\frac{\partial}{\partial t}\rho e_e + \nabla\cdot(\rho e_e\mathbf{u}) + \nabla\cdot\mathbf{q}_e + \mathrm{P}_e : \nabla\mathbf{u} &= \gamma_{ei}(T_i - T_e) + c\sigma_a(E_r - at_e^4) + \dot{S}_e\\
\frac{\partial}{\partial t}\rho E + \nabla\cdot[(\rho E| + \mathrm{P})\cdot\mathbf{u}] + \nabla\cdot(\mathbf{q}_e + \mathbf{q}_i) &= c\sigma_a(E_r - aT_e^4) - \frac{1}{3}\mathbf{u}\cdot\nabla E_r + \dot{S}_i + \dot{S}_e\\
\frac{\partial}{\partial t}E_r + \frac{4}{3}\nabla\cdot(\mathbf{u}E_r) - \nabla\cdot(\kappa\nabla E_r) &= -c\sigma_a(E_r - aT_e^4) + \frac{1}{3}\mathbf{u}\cdot\nabla E_r
\end{aligned}$$

where ρ is density, $\mathbf{u}$ is fluid velocity, $\mathrm{P} = \mathrm{P}_e + \mathrm{P}_i$ is the pressure tensor, e_j is the specific heat for the jth constituent, T_j is temperature, γ_{ie} is the electron-ion coupling relation, $\mathbf{q}_j$ are heat fluxes, σ_a is the absorption opacity, E_r is the frequency averaged energy in the radiation field and κ is the radiation diffusion coefficient.

In the one dimensional case, neglecting source terms and spatial dependence, and considering only the relaxation of the temperatures of the various constituents allows us to reduce these equations to

$$\frac{\partial E_r}{\partial t} = c\sigma_a(aT_e^4 - E_r) \tag{2.1}$$

$$\frac{\partial \rho e_e}{\partial t} = \gamma_{ei}(T_i - T_e) - c\sigma_a(aT_e^4 - E_r) \tag{2.2}$$

$$\frac{\partial \rho e_i}{\partial t} = \gamma_{ei}(T_e - T_i). \tag{2.3}$$

Because of the simplifications just mentioned, these equations describe the time evolution of the equilibration between the respective energies of the system's constituents (that is, at late times, we expect that the temperatures will equilibrate to some final E_f). RAGE's existing structure forbids an easy implementation of a fully implicit scheme. For part of the integration, RAGE instead uses a semi-implicit, operator split scheme on a linearized version of (2.1-2.3) with nonlinearities lagged or iterated. Numerous linearizations are possible. We consider

$$\frac{\partial T_r}{\partial t} = \frac{T_e - T_r}{2\tau_R} \tag{2.4}$$

$$\frac{\partial T_e}{\partial t} = f_2 \frac{T_r - T_e}{2\tau_R} + f_1 \frac{T_i - T_e}{2\tau_P} \tag{2.5}$$

$$\frac{\partial T_i}{\partial t} = \frac{T_e - T_i}{2\tau_P}, \tag{2.6}$$

where $\tau_P = \frac{\rho C_{v,i}}{2\gamma_{ei}}$, $\tau_R = \frac{1}{2c\tilde{\sigma}_a}$, $f_1 = \frac{C_{v,i}}{C_{v,e}}$, $f_2 = \frac{4aT_r^3}{\rho C_{v,e}}$ and we have used $E_r = aT_r^4$. For the sake of this analysis, we will treat each of these coefficients as constants in consistency with RAGE's semi-implicit operator split integration scheme. If this system is represented in matrix form, i.e.

$$\frac{\partial}{\partial t}\mathbf{x} = \mathbf{A}\mathbf{x},$$

for $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{A} \in \mathbb{R}^3 \times \mathbb{R}^3$, then

$$\mathbf{A} = \begin{pmatrix} -\frac{1}{2\tau_R} & \frac{1}{2\tau_R} & 0 \\ \frac{f_2}{2\tau_R} & -\left(\frac{f_2}{2\tau_r} + \frac{f_1}{2\tau_P}\right) & \frac{f_1}{2\tau_P} \\ 0 & \frac{1}{2\tau_P} & -\frac{1}{2\tau_P} \end{pmatrix},$$

which has non-trivial eigenvalues

$$\lambda_j = -\frac{\tau_P + f_2\tau_P + \tau_R + f_1\tau_R \pm \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}}{4\tau_P\tau_R}.$$

Remark 1. *The nontrivial eigenvalues λ_j are real, distinct and negative.*

The proof appears in the appendix. We are now prepared for

Definition 1. *If*

$$\tau_j = -\frac{1}{\lambda_j},$$

then we say that τ_j is the relaxation time associated with eigenvalue λ_j.

Remark 2. *Note that, because of the validity of Remark 1, $\tau_j \in \mathbb{R}^+$.*

We might initially feel uncomfortable with the above definition as the equilibration of the three temperatures which we have described is ultimately non-linear and not precisely exponential as we would anticipate if our coefficients τ_P, τ_R, f_1, f_2 were actually constant. The quantity τ_j is still valuable, however, in providing an instantaneous timescale for the system. The reader is also reminded that the τ_j's are themselves functions of the three temperature vector $\mathbf{x}$. Our goal is to employ these relaxation times in the selection of a time stepsize. It may be helpful to demonstrate the merits of utilizing relaxation times as timescales for the system. To motivate our discussion, we introduce the concept of stiffness.

2.1. **Criteria for Stiffness.** In general, stiffness is a difficult property of ODEs to define. In our case, it is appropriate to quantify stiffness with

Definition 2. *If the quantity*

$$\left|\frac{\overline{\lambda}}{\underline{\lambda}}\right|,$$

where $\overline{\lambda}$ and $\underline{\lambda}$ are the largest and smallest eigenvalues of the system respectively, becomes large we say that the system of ODEs has become stiff.

Clearly Definition 2 cannot apply to all systems of ODEs (some obviously stiff systems of equations contain only a single eigenvalue), but the definition is a safe one for our purposes. The ratio effectively measures the difference in timescales of various relaxing components of the system, a common signature of stiff systems. As the ratio becomes large, the system is relaxing on two very different timescales and care must be taken to ensure proper time stepsize selection both to resolve the rapid relaxation sufficiently and to integrate the equations efficiently.

In our particular case

$$\left|\frac{\overline{\lambda}}{\underline{\lambda}}\right| = \frac{\tau_P + f_2\tau_P + \tau_R + f_1\tau_R + \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}}{\tau_P + f_2\tau_P + \tau_R + f_1\tau_R - \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}},$$

which becomes large for any of the following cases:

- $\tau_P \to 0, \infty$,
- $\tau_R \to 0, \infty$,
- $f_1, f_2 \to \infty$.

We omit the proof. Depending on the forms chosen for $\gamma_{e,i}$ and $C_{v,e}$, $C_{v,i}$, these conditions become equivalent to large temperature differences between the radiation and electrons or the electrons and ions.

It has been well-established in the literature that substantially stiff systems require very small timesteps to keep integration errors within tolerance. It can be shown that $\tau_j \to 0$ with diminishing τ_P and τ_R as well as $f_1, f_2 \to \infty$, which would be a desirable property of a time stepsize selection scheme.

2.2. **The New Scheme.** The time stepsize selection scheme we have implemented in RAGE is

$$\Delta t_R^{n+1} = \max\left\{\lambda_R\tau_R, \min_j\left[\Delta t^n\left(\frac{\eta_{target,R}}{\eta_j}\right)^{1/2}\right]\right\},$$

and

$$\Delta t_P^{n+1} = \max\left\{\lambda_P\tau_P, \min_j\left[\Delta t^n\left(\frac{\eta_{target,P}}{\eta_j}\right)^{1/2}\right]\right\},$$

with

$$\eta_j = \frac{|T_j^n - T_j^{n+1}|}{T_j^n},$$

and $j \in \{i, e, r\}$. The quantity $\min_j\left[\Delta t_{old}\left(\frac{\eta_{target,p}}{\eta_j}\right)^{1/2}\right]$ is a well-established time step size control from the literature (see e.g. Knoll, et al. 2001, their equation 11), with $\eta_{target,p}$ a carefully chosen constant ($p \in \{P, R\}$). The effect of this scheme is to enforce a lower bound on the stepsize based on the relaxation times of the couplings. These time stepsize recommendations are calculated respectively the

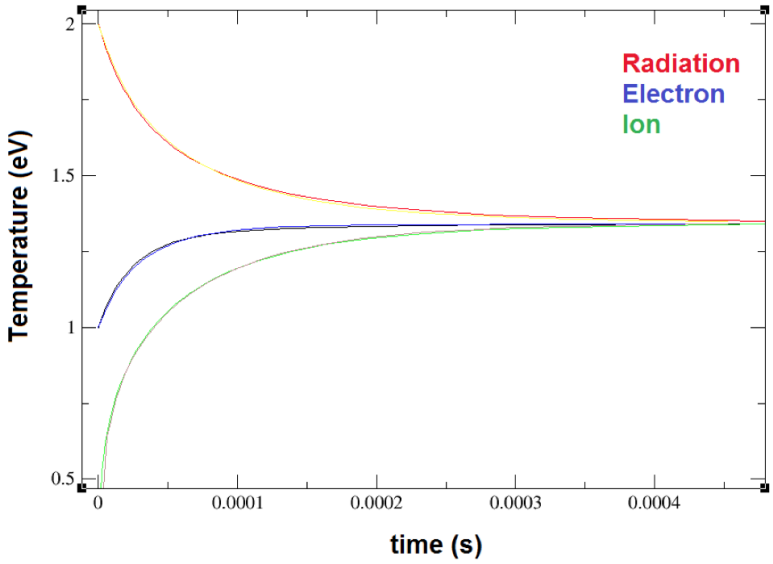


FIGURE 1. Integration results for both the old and new stepsize selection scheme for the 3t_eir_relax.test problem. Differences in integration are barely discernable by eye. A color version of this figure appears in the electronic copy. Image generated with xshow.

radiation and plasma sections of the code. This timestep option may be activated by using "improved_tsc = .true." in a RAGE input deck. This option also turns off the "3tFORCE" and "rad zero" initial time stepsize recommendations in favor for an initial stepsize recommendation based on the relaxation times. The algorithm has four adjustable parameters, $\lambda_R, \lambda_P, \eta_{target,R}, \eta_{target,P}$ which may also be specified in the input deck.

We tested the speed of the newer algorithm, expecting that significantly larger timesteps would be employed in the integration. We constructed an ensemble of relaxation problems to probe the efficiency of the algorithm in different initial conditions. The magnitudes of the initial temperatures were permuted across the test problems with varying degrees of stiffness. In this way, we hoped to sufficiently sample the space the all possible initial conditions. Problems were run on a 300×300 domain and 16 processors on Moonlight. The wall times and number of cycles (steps) employed by the old and our new method appear in Table 1. We note that for the vast majority of relaxation problems, number of cycles and wall times are diminished by a factor of 1/2 in the new method.

The integration results differed from the results of the RAGE gold standard for the test problem. These differences were barely discernible by eye when the two integrations are plotted on top of each other (see Figure 1). We assert that for a large number of three temperature plasma problems, the improved timestep scheme will give appropriate results at a fraction of the wall time without sacrificing significant physics.

	Old Timestep Scheme		**New 3T Stepsize Controls**	
Test Problem	Cycles	Wall Time (s)	Cycles	Wall Time (s)
3t_ei_relax	64	56	64	56
3t_eir_relax	64	16	29	10
3t_relax_01	64	115	29	54
3t_relax_02	64	124	33	62
3t_relax_03	64	113	26	48
3t_relax_04	64	124	26	53
3t_relax_05	64	122	33	59
3t_relax_06	64	119	29	53
3t_relax_07	64	107	27	49
3t_relax_08	64	111	32	58
3t_relax_09	64	113	32	57
3t_relax_10	64	117	27	51
3t_relax_11	64	117	27	51
3t_relax_12	64	112	32	59

TABLE 1. Efficiency comparisons between the old stepsize scheme and our recently implemented stepsize controls. The new scheme efficiently integrates in roughly half the number of cycles and wall time in nearly every test case in relax.suite. Test problems were run on Moonlight on 16 processors each and on 300×300 grids.

More elaborate time step size selection schemes were considered which would select even more conservative time steps while still circumventing RAGE's tendency to select unnecessarily small step sizes. RAGE's complex structure prevented very involved implementation, though we are confident that other step size controls could be installed with additional time. For the remainder of this report, we will discuss some additional ideas which might be of value to RAGE developers.

2.3. **A Suggested Time Step Size Selection Scheme.** We motivate our discussion by introducing the toy system

$$\frac{\partial T_r}{\partial t} = \frac{T_e - T_r}{2\tau_R}, \tag{2.7}$$

$$\frac{\partial T_e}{\partial t} = \frac{T_r - T_e}{2\tau_R} + \frac{T_i - T_e}{2\tau_P}, \tag{2.8}$$

$$\frac{\partial T_i}{\partial t} = \frac{T_e - T_i}{2\tau_P}, \tag{2.9}$$

which has two distinct eigenvalues

$$\lambda_j' = \frac{-(\tau_R + \tau_P) \pm \sqrt{(\tau_R + \tau_P)^2 - 3\tau_R\tau_P}}{2\tau_R\tau_P},$$

where τ_P and τ_R are the relaxation times for the ion and radiation coupling respectively and we have included a prime on λ_j to distinguish them for the aforementioned eigenvalues. The eigenvalues λ_j' correspond to relaxation times for the

global system via

$$\tau'_j = -\frac{1}{\lambda'_j}$$

We say $\tau'_- = \min_j\{\tau'_j\}$ is the minimum relaxation time for the entire system. We are now prepared to make an important claim.

Claim 1. *The global, minimum relaxation time τ'_- is less than the individual relaxation times for the coupled relaxation ships within the system, i.e.*

$$\tau'_- < \tau_R$$

and

$$\tau'_- < \tau_P$$

for every $\tau'_-, \tau_P, \tau_R \in \mathbb{R}^+$.

Proof. The validity of this claim may be demonstrated by means of a simple algebraic argument. Because $\tau_P, \tau_R \in \mathbb{R}^+$ we know that

$$\tau_R\tau_P > 0 \Rightarrow 3\tau_R\tau_P > 0 \Rightarrow 2\tau_R\tau_P - 3\tau_R\tau_P > -2\tau_R\tau_P.$$

Adding $\tau_R^2 + \tau_P^2$ to both sides preserves the inequality yielding (after selective factoring)

$$(\tau_R + \tau_P)^2 - 3\tau_R\tau_P > (\tau_P - \tau_R)^2 \Rightarrow \sqrt{(\tau_R + \tau_P)^2 - 3\tau_R\tau_P} > \tau_P - \tau_R, \tag{2.10}$$

because of the monotone nature of the square root function. It is now easy to see that

$$\tau_R + \tau_P + \sqrt{(\tau_R + \tau_P)^2 - 3\tau_R\tau_P} > 2\tau_P \Rightarrow \frac{2\tau_R\tau_P}{\tau_R + \tau_P + \sqrt{(\tau_R + \tau_P)^2 - 3\tau_R\tau_P}} < \tau_R,$$

or that

$$\tau'_- = \frac{1}{\lambda_-} = \frac{2\tau_R\tau_P}{\tau_R + \tau_P + \sqrt{(\tau_R + \tau_P)^2 - 3\tau_R\tau_P}} < \tau_R.$$

Because τ'_- is symmetric in τ_R and τ_P, one can easily show $\tau'_- < \tau_P$ by exploiting the fact that $\tau_R^2 - 2\tau_R\tau_P + \tau_P^2$ may also be factored like $(\tau_R - \tau_P)^2$ in addition to the factoring which appears in (2.10). □

This property of this toy system is also evidenced in our linearized equations (2.4-6). The (approximate) Jacobians of the individual couplings for this system are

$$\mathcal{L}_1 = \begin{pmatrix} -\frac{1}{2\tau_R} & \frac{1}{2\tau_R} \\ \frac{f_2}{2\tau_R} & -\frac{f_2}{2\tau_R} \end{pmatrix},$$

$$\mathcal{L}_2 = \begin{pmatrix} \frac{f_1}{2\tau_P} & -\frac{f_1}{2\tau_P} \\ -\frac{1}{2\tau_P} & \frac{1}{2\tau_P} \end{pmatrix},$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are the approximate Jacobians for the radiative and plasma coupling steps respectively. Each Jacobian has one non-trivial eigenvalue which correspond to a relaxation time. These are

$$\tau_{rad} = \frac{2\tau_R}{1 + f_2}, \quad \tau_{plas} = \frac{2\tau_P}{1 + f_1}.$$

We note that τ_P and τ_R are no longer the relaxation times of the system but $\tau_{rad} \to \tau_R$ and $\tau_{plas} \to \tau_P$ when $f_1, f_2 \to 1$.

Claim 2. *The minimum relaxation time of the entire system τ_- is smaller than the relaxation times of the individual components* $\left(\frac{2\tau_R}{1+f_2}, \frac{2\tau_P}{1+f_1}\right)$.

Proof. We first prove that $\tau_- < \frac{2\tau_R}{1+f_2}$. Because $f_1, f_2, \tau_P, \tau_R \in \mathbb{R}^+$, it is clear that

$$4f_1f_2\tau_P\tau_R > 0.$$

We now add and subtract $\phi = \tau_R^2 + 2f_1\tau_R^2 + f_1^2\tau_R^2 + \tau_P^2 + 2f_2\tau_P^2 + f_2^2\tau_P^2 - 2\tau_P\tau_R - 2f_1\tau_P\tau_R - 2f_2\tau_P\tau_R$ from the LHS. The resulting inequality can be shown to be equivalent to

$$(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2) - ((1 + f_2)\tau_P - \tau_R(1 + f_2))^2 > 0.$$

Moving the last term on the LHS to the RHS and taking the square root implies that

(2.11)
$$\sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} > |(1 + f_2)\tau_P - \tau_R(1 + f_2)|.$$

We can prove that $\tau_- < \frac{2\tau_R}{1+f_2}$ by considering the case in which

$$|(1 + f_2)\tau_P - \tau_R(1 + f_2)| = (1 + f_2)\tau_P - \tau_R(1 + f_2).$$

In this case, (2.11) can be shown to be equivalent to

$$\sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} > (2 - 1)(1 + f_2)\tau_P - \tau_R(1 + f_2).$$

Rearranging gives

$$\tau_P(1+f_2)+\tau_R(1+f_1)+\sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} > 2(1+f_2)\tau_P,$$

which implies that

$$\frac{4\tau_P\tau_R}{\tau_P(1 + f_2) + \tau_R(1 + f_1) + \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}} < \frac{2\tau_R}{1 + f_2},$$

or that

$$\tau_- < \frac{2\tau_R}{1 + f_2}.$$

We can next show that $\tau_- < \frac{2\tau_P}{1+f_1}$ by considering the remaining case

$$|(1 + f_2)\tau_P - \tau_R(1 + f_2)| = -(1 + f_2)\tau_P + \tau_R(1 + f_2).$$

It may be shown that (2.11) then becomes

$$\sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} > -(1 + f_2)\tau_P + (2 - 1)\tau_R(1 + f_2).$$

Rearranging the expression gives

$$\tau_P(1+f_2)+\tau_R(1+f_1)+\sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} > 2\tau_R(1+f_2),$$

which is to say that

$$\frac{4\tau_P\tau_R}{\tau_P(1 + f_2) + \tau_R(1 + f_1) + \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}} < \frac{2\tau_P}{1 + f_1},$$

and that

$$\tau_- < \frac{2\tau_P}{1 + f_1},$$

as claimed. □

The above claim demonstrates the advantage of using τ_- over the individual relaxation times for the system to determine the next step size. Though the claim is instructive, it may be of interest to the reader to know under what conditions $\tau_- < \tau_P, \tau_R$, quantities which are computed during RAGE's integration. This would require simplifying the inequality

$$\frac{4\tau_P\tau_R}{\tau_P(1+f_2)+\tau_R(1+f_1)+\sqrt{(\tau_P+f_2\tau_P+\tau_R+f_1\tau_R)^2-4(\tau_P\tau_R)(1+f_1+f_2)}} < \tau_P.$$

After routine manipulations, we find that

$$\sqrt{(\tau_P+f_2\tau_P+\tau_R+f_1\tau_R)^2-4(\tau_P\tau_R)(1+f_1+f_2)} > 3\tau_R-\tau_P(1+f_2)-f_1\tau_R,$$

which is equivalent to requiring

$$\begin{cases} f_1 > \frac{\tau_P+f_2\tau_P-2\tau_R}{\tau_P-2\tau_R} & \text{if } \tau_P \neq 2\tau_R \\ f_2 > \frac{-\tau_P+2\tau_R}{\tau_P} & \text{if } \tau_P = 2\tau_R \end{cases}.$$

A similar routine may be carried out to find constraints on f_1, f_2 to achieve $\tau_- < \tau_R$. We need

$$\begin{cases} f_2 < \frac{2\tau_P-\tau_R-f_1\tau_R}{2\tau_P-\tau_R} & \text{if } \tau_P \neq \tau_R/2 \\ f_1 > \frac{2\tau_P-\tau_R}{\tau_R} & \text{if } \tau_P = \tau_R/2 \end{cases}.$$

These requirements describe a somewhat complicated region in a 4-D parameter space. For concreteness sake, we present for your consideration

Example 1. *Note that if $\tau_P = \tau_R = \tau$, we have*

$$\tau_+ = \frac{4\tau^2}{2\tau} = 2\tau,$$

and

$$\tau_- = \frac{2\tau}{1+f_1+f_2},$$

which is less than the relaxation time of the individual elements of the system if $f_1 + f_2 > 1$.

It would appear that it would be advantageous to have a time stepsize selection scheme which is based on the global relaxation times for the system τ_- and τ_+ even over a scheme based τ_R and τ_P as we have recently implemented. Our suggested timestep scheme is

$$\Delta t_{new} = \min\left\{\max\left[\lambda\tau_-, \min_j\left[\Delta t_{old}\left(\frac{\eta_{target,R}}{\eta_j}\right)^{1/2}\right]\right], \alpha\tau_+\right\},$$

where λ, α are carefully chosen constants. Though this algorithm has not been tested directly in the RAGE environment, we have demonstrated some of its merits externally in a code which we wrote (iSILDR: Semi-implicit Integrator of the Linearized Differential equations of RAGE) to mimic RAGE's integration scheme. The details of the integration algorithm may be found in the Appendix. We did 100 integrations with this algorithm with each initial temperature $T_{0,j}$ taken from a uniform distribution from 0 to 10. In this way, we hoped to adequately sample the space of all possible initial conditions. The radiation-electron coupling was set stronger than the ion-electron coupling. A typical integration from these runs appears in Figure 2.

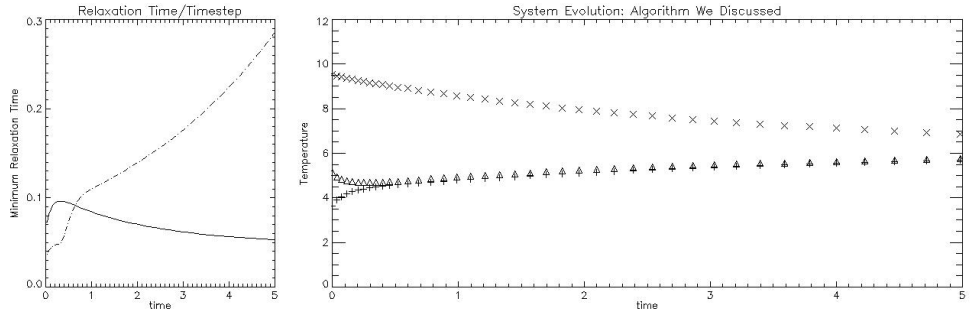


FIGURE 2. *(Left)* Minimum Relaxation Time τ_- is plotted as a solid line. The dotted line indicates the timestep selected by the algorithm. Each quantity is represented here as a function of time t. *(Right)* Temperatures as a function of time. Markers represent data points computed in the discrete integration. Note that the time stepping algorithm exhibits the pleasing property that Δt gets larger as the system relaxes (i.e. markers get farther apart).

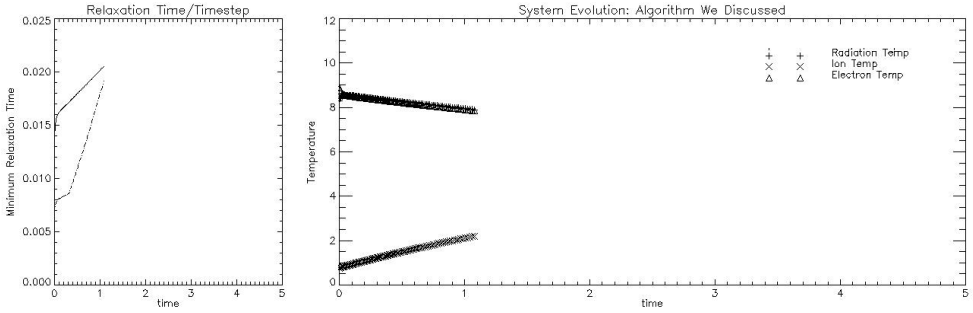


FIGURE 3. A scenario where our time stepping scheme appears to integrate inefficiently. Note that if the strongly coupled pair starts off close to each other, the simulation will choose an appropriately small timestep to resolve the quicker relaxation. The integration, however, grinds on for a much longer time than we would like (possibly because of the size of η_{target}).

We note that the integration scheme possesses the pleasing property that Δt becomes large as the system equilibrates. Note that the user has the ability to adjust the coarseness of the integration by setting η_{target}.

2.4. **A Timestepping Scheme for Stiff Scenarios.** Though the above method works well for a variety of circumstances, the scheme we have just suggested experiences difficulties with initial conditions which cause the equations to become stiff. This is particularly evident in the integration shown in Figure 3. The integration appears to grind on far longer through an essentially featureless relaxation. One

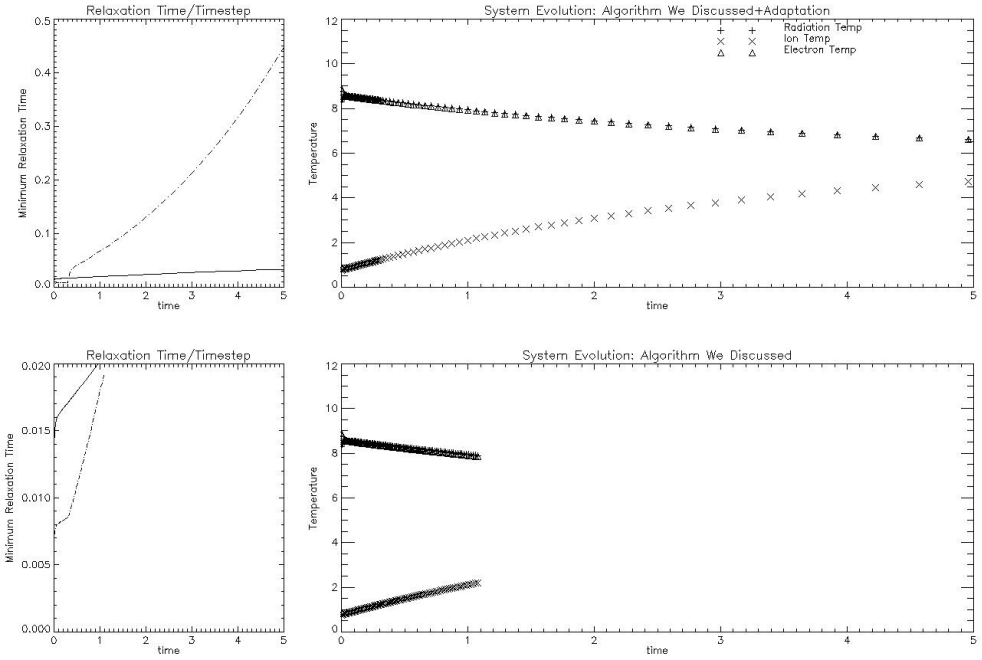


FIGURE 4. Comparison of (1.1) *(below)* and the adjusted algorithm (see section 2) *(above)* for identical initial conditions. The two schemes employ the same number of integration steps though the adjusted algorithm integrates the long, features, final relaxation much more efficiently. Note that within 100 timesteps the algorithm in (1.1) will not make it past about 1.1 time units. The adjusted algorithm integrates well beyond 5 time units with the same number of steps.

might attempt to adjust η_{target} to compensate for the slow integration, and such a solution certainly has merit. We sought, however, a scheme which would work for general initial conditions, requiring no *a priori* knowledge from the user of the stiffness of the equations.

We noticed that for virtually all of the simulations $\tau_- > \min_j \left[\Delta t_{old} \left(\frac{\eta_{target}}{\eta_j}\right)^{1/2}\right]$ for small t. We added the following "if" statement following the computation of the timestep: If $\tau_- < \min_j \left[\Delta t_{old} \left(\frac{\eta_{target}}{\eta_j}\right)^{1/2}\right]$ then $\Delta t = \alpha\tau_+$. I set α to 1.0. The results of this simple implementation follow in Figures 4 and 5.

It appears that when the recommended timestep exceeds the minimum relaxation time of the system (i.e. (1.1) gives $\Delta t \neq \tau_-$), the relaxing system may be safely integrated with a significantly larger timestep than that recommended by our initial scheme. In this experiment, we have used a timestep which is on the order of the maximum relaxation time for the system. While this claim will need to be verified with appropriate error analysis and though non-linearities are not fully converged

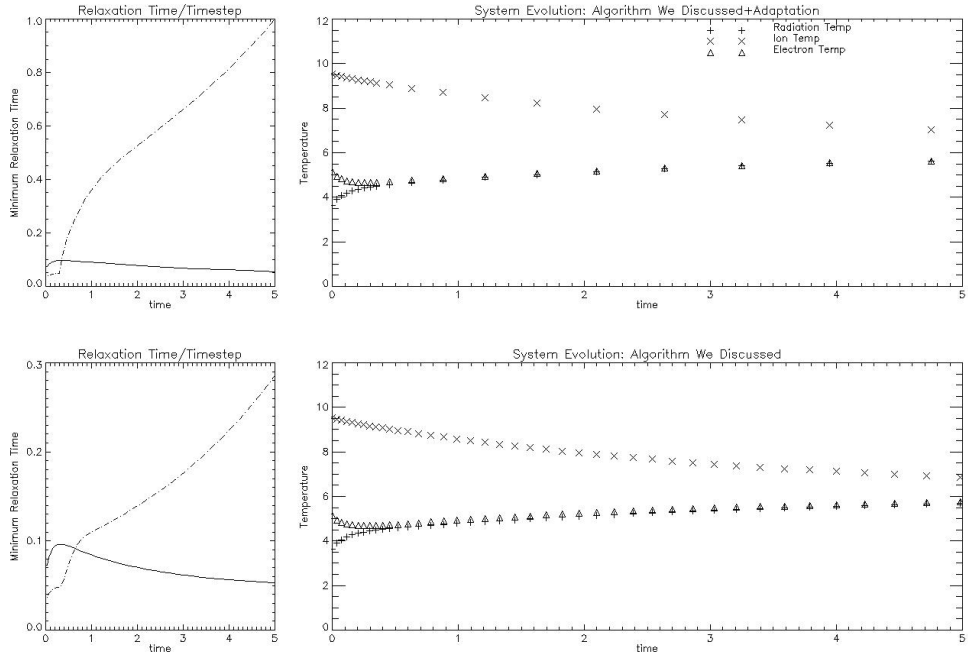


FIGURE 5. Same as Figure 4. Notice that the new adjusted scheme picks up speed only after complex features in the system have been resolved (i.e. after the "stiffer" coupling has relaxed).

in our integration scheme, it might be worthy of note that in every experiment the "accelerated" integration in this new scheme takes place only following exotic features (maxima, minima, etc.).

We make one additional observation.

Remark 3. *In a relaxing system, the usual stepsize selection scheme* ($\Delta t^{n+1} = \Delta t^n \left(\frac{\eta_{target}}{\eta}\right)^{1/2}$, *hereafter the "usual" stepsize scheme) becomes greater than the minimum relaxation time for the system* τ_-, *the stiffer coupling has relaxed. When this same stepsize criteria recommends larger stepsizes than the maximum relaxation time* τ_+, *the entire system has relaxed.*

This observation has some nice implications. Because there are only three temperatures relaxing, as soon as the usual timestep exceeds τ_-, all local extrema have been integrated and a larger step size may be used (on the order of τ_+). As soon as the usual stepsize scheme recommends stepsizes larger than τ_+, very large step sizes may be employed pending no possibility of sourcing. This is also convenient as the relaxation times and the usual scheme can together operate as a type of indicator that specific events have taken place in the evolution of the system.

The reader may well wonder if the usual stepsize recommendation exceeding the minimum relaxation time actually signals a special benchmark in the relaxation of the system. To test this idea, we carried out 2000 integrations with iSILDR with initial temperatures randomly selected from uniform distributions. For each

set of initial conditions, we varied λ on the interval $[0, 2.0]$. Because λ represents a threshold which the recommendation of the usual stepsize scheme must exceed, larger values of λ correspond to later times at which the integration will make the switch between timesteps on the order of the minimum relaxation time τ_- and steps close to τ_+, the maximum relaxation time. The results of this error analysis are shown in Figure 6. Lines higher in L2 error norms correspond to sets of integrations carried out on successively stiffer initial conditions. This increase in error is thought to largely be due to the fixed order of the operator split and the unconverged nonlinearities in our semi-implicit scheme.

We note that error drops substantially as the "jump-off point" λ becomes larger but that most improvement in L2 norm of the error is gained by the time the usual scheme recommends stepsizes which exceed about 1.0 minimum relaxation time. Figure 6 illustrates that little benefit is gained by integrating with very small time stepsizes after this point (i.e. after the stiffer coupling has relaxed). Averages across all levels of stiffness indicate that over 95% of the improvement in error is already gained at this point. Future time step size controls for RAGE could exploit this fact in step size selection.

3. Operator Split Methods

We first begin with

Claim 3. *An operator split algorithm composed of two fully implicit schemes is unconditionally stable.*

A proof appears in the appendix. While this may offer comfort for the use of operator split methods with fully implicit substeps, it can be shown that issues with non-converged nonlinearities (as are present in our semi-implicit scheme) are attenuated by operator splitting. Notwithstanding the former argument, integrating the three temperature equations without operator split methods would be optimal.

3.1. Results from Runs Comparing Split Methods.

Though operator split methods have been studied in general, and though it might appear that higher order split methods are always advantageous, the true merits of an operator split method rely heavily on the nature of the problem at hand (see Kozlov et. al. 2003). In this subsection, we seek to determine the method best suited for the three temperature relaxation problem.

Five different integration schemes were employed on a stiff problem simultaneously in iSILDR: both variations of Yanenko (equivalently "Lie" in this case) and Strang splitting in addition to a throttled integration for error calculation purposes. The equations were stiff in the electron-ion coupling. Our purpose in this experiment was to ascertain the relative integrity of split methods in a stiff system in tracing out a local extremum. The experimental results for the electron temperature T_e are displayed in Figure 7.

Somewhat surprisingly, first order Lie splitting which performed the stiff integration first was more accurate than all other integration schemes, including those of higher order. In each case, integrating the stiff coupling first provides better results. In an attempt to investigate this more fully, I carried out 100 additional integrations with varying levels of stiffness (see Figure 8). For clarity only the electron temperature is shown though similar plots might be generated for the other plasma

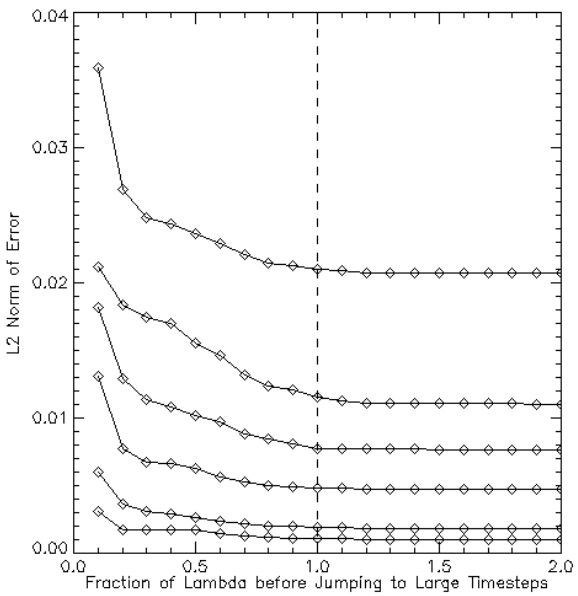


FIGURE 6. L2 norm of the error of integrations as a function of the fraction of the minimum relaxation time the Ryder-Knoll timestep must be greater than before step size is automatically chosen to be the larger relaxation time. Each diamond represents the average L2 norm of the error for 50 integrations. Diamonds connected by lines represent runs which shared the same set of initial conditions. Higher lines on the plot represent successively stiffer initial conditions. The dotted vertical line indicates simulations which "jumped" to larger time stepsizes after exactly 1.0 relaxation time. On average, 95.7% of the improvement in error gained by waiting to jump until the Ryder-Knoll timestep becomes very large is gained before $\lambda = 1.0$. $\lambda = 1.0$ appears to be an appropriate place to "jump" regardless of the stiffness of the system.

constituents. It appears that for this test problem, integrating the stiff section first consistently produces better results over a variety of stiff initial conditions.

RAGE could benefit from an implementation where the stiffer component is always intergrated first. The current operator split method in RAGE is actually fortuitous as the relaxation times for the plasma coupling is generally much smaller than the relaxation times for the radiation. This is accomodated well by RAGE's design in always integrating the plasma coupling before the radiation. In circumstances where this trend does not hold, integrating the stiffer coupling first would be beneficial.

Error analyses were also carried out for Strang splitting. Though Strang splitting did not offer the best results out of the methods considered, it should be noted that Strang splitting with the stiff component integrated between half steps of non-stiff components faired best over a variety of stiff initial conditions.

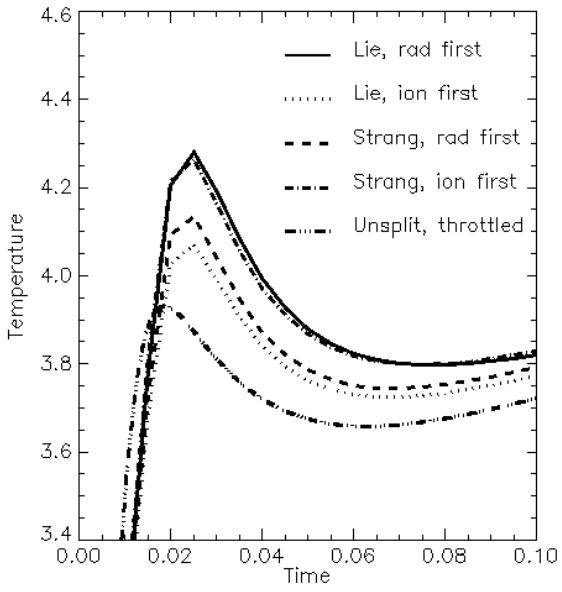


FIGURE 7. Five separate integrations of electron temperature in a 3-T system with a stiff ion-electron coupling. The throttled timestep integration is the most accurate of those depicted.

4. Conclusion

In this report, we have outlined some of the time step size selection scheme difficulties in the RAGE code and have outlined our improvements to it. We have reduced wall time for large, simple three temperature relaxation problems by a factor of 1/2 without seriously compromising the accuracy of the integration. The new scheme has four adjustable parameters which can be fine-tuned to meet the accuracy and speed needs of a given integration.

We have also outlined additional ideas for time step size recommendation based on the relaxation times for the entire system. We have presented the important result that the minimum relaxation time for the entire system is smaller than the relaxation times of the individual couplings. We have also suggested an additional scheme which might work well for stiff initial conditions, based on our observation that when a common time step control recommends timesteps larger than the current minimum relaxation time, it appears to be an indication that the stiffer coupling has relaxed. We have briefly discussed the merits of different operator split methods and have recommended Lie Splitting with the stiffer coupling also integrated first.

Acknowledgements

I would like to thank Katie Gosmeyer for fruitful conversations and for her very helpful tutoring of me in IDL. A very special thanks is due to our excellent mentor Tom Masser whose insights guided this work. We are also grateful to Scott Runnels for orchestrating the workshop and Tena Jenkins for her help throughout the

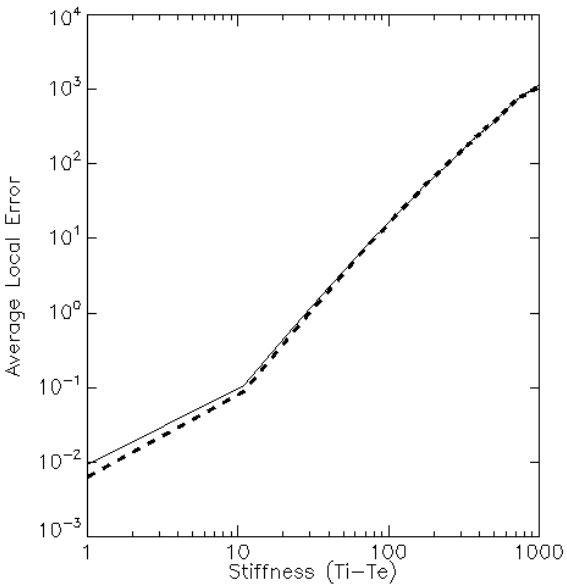


FIGURE 8. Average Local Errors for two integration methods and over 100 integrations. The solid line represents Lie Splitting with the non-stiff component integrated first. The dotted line is Lie Splitting with the stiff coupling integrated first. Errors for Lie operator splitting with the stiff component integrated first outperform the alternate integration for all levels of stiffness.

summer. We would also like to thank John Wöhlbier for his help at the beginning of the summer. We also thank Los Alamos National Laboratory for this singular summer experience. This work was funded by the ASC Hydrodynamics Project.

APPENDIX

4.1. Proof of the Real, Distinct, Non-negative Nature of the Eigenvalues λ_j.

Proof. We first show that the eigenvalues are real or that $D = (\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2) > 0$. Observe that

$$\begin{aligned} D &= \tau_P^2 + 2f_1\tau_P^2 + f_1^2\tau_P^2 - 2\tau_P\tau_R - 2f_1\tau_P\tau_R - 2f_2\tau_P\tau_R \\ &\quad +2f_1f_2\tau_P\tau_R + \tau_R^2 + 2f_2\tau_R^2 + f_2^2\tau_R^2 \\ &= (\tau_P + f_1\tau_P - \tau_R - f_2\tau_R)^2 + 3f_1f_2\tau_P\tau_R > 0 \end{aligned}$$

because $\tau_P, \tau_R, f_2, f_1 > 0$. Because determinant $D > 0$, $D \neq 0$ so the eigenvalues are *distinct*.

We next show that the non-trivial eigenvalues λ_i are negative. It is clear that

$$\lambda_1 = -\frac{\tau_P + f_2\tau_P + \tau_R + f_1\tau_R - \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)}}{4\tau_P\tau_R} < 0$$

if $\tau_P, \tau_R > 0$. It is also apparent that

$$\begin{aligned} \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} &< \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2} \\ &= \tau_P + f_2\tau_P + \tau_R + f_1\tau_R, \end{aligned}$$

which only holds (again) if all the variables here are positive. It is now clear that

$$\tau_P + f_2\tau_P + \tau_R + f_1\tau_R + \sqrt{(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)^2 - 4(\tau_P\tau_R)(1 + f_1 + f_2)} < 0$$

which implies that $\lambda_2 < 0$ as asserted. □

4.2. **Algorithm Details for iSILDR.** We seek to solve the system

$$\frac{\partial T_r}{\partial t} = \frac{T_e - T_r}{2\tau_R}, \tag{4.1}$$

$$\frac{\partial T_e}{\partial t} = -f_2\frac{T_e - T_r}{2\tau_R} - f_1\frac{T_e - T_i}{2\tau_P}, \tag{4.2}$$

$$\frac{\partial T_i}{\partial t} = \frac{T_e - T_i}{2\tau_P}, \tag{4.3}$$

in a manner which is consistent will sufficiently mimic RAGE's integration scheme. RAGE uses a Lie operator split method and integrates the plasma coupling with a semi-implicit Euler scheme. Integration of the radiation coupling is more complicated, involving the iteration of nonlinearities. For the purposes of our experiment, we will use a simple Lie operator split method with a first order semi-implicit Euler scheme (lagging nonlinear coefficients).

We now develop the formulae. We can express the above equation set in matrix form:

$$\frac{\partial}{\partial t}\mathbf{T} = \mathcal{L}\mathbf{T},$$

where $\mathbf{T} \in \mathbb{R}^3$ and $\mathcal{L}$ is the appropriate 3×3 matrix. Because $\mathcal{L}_1 = \begin{pmatrix} -\frac{1}{2\tau_R} & \frac{1}{2\tau_R} & 0 \\ \frac{f_2}{2\tau_R} & -\frac{f_2}{2\tau_R} & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $\mathcal{L}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\frac{f_1}{2\tau_P} & \frac{f_1}{2\tau_P} \\ 0 & \frac{1}{2\tau_P} & -\frac{1}{2\tau_P} \end{pmatrix}$ are such that $\mathcal{L}_1 + \mathcal{L}_2 = \mathcal{L}$ we can write the equation set as

$$\frac{\partial}{\partial t}\mathbf{T} = (\mathcal{L}_1 + \mathcal{L}_2)\mathbf{T}.$$

We will approximate the radiation step with an implicit method (necessarily lagging the nonlinear coefficients) with

$$\frac{T_r^{n+1} - T_r^n}{\Delta t} = \frac{1}{2\tau_R^n}\left(T_e^{n+1} - T_r^{n+1}\right), \tag{4.4}$$

$$\frac{T_e^{n+1} - T_e^n}{\Delta t} = -\frac{f_2^n}{2\tau_R^n}\left(T_e^{n+1} - T_r^{n+1}\right), \tag{4.5}$$

which, solved for T_r^{n+1} and T_e^{n+1}, is

$$T_r^{n+1} = \frac{\Delta t T_e^n + \Delta t f_2^n T_r^n + 2T_r^n\tau_R}{\Delta t + \Delta t f_2^n + 2\tau_R}, \tag{4.6}$$

$$T_e^{n+1} = \frac{\Delta t T_e^n + \Delta t f_2^n T_r^n + 2T_e^n\tau_R}{\Delta t + \Delta t f_2^n + 2\tau_R}. \tag{4.7}$$

Updating temperatures in the electron-ion coupling step are carried out similarly:

$$T_e^{n+1} = \frac{\Delta t T_e^n + \Delta t f_1^n T_i^n + 2\tau_P^n T_e^n}{\Delta t + f_1^n\Delta t + 2\tau_P^n}, \tag{4.8}$$

$$T_i^{n+1} = \frac{\Delta t T_e^n + \Delta t f_1^n T_i^n + 2\tau_P^n T_i^n}{\Delta t + f_1^n\Delta t + 2\tau_P^n}. \tag{4.9}$$

We first update coefficients and use

$$T^{n+1} = T^n + \Delta t, \tag{4.10}$$

$$T_r^{n+1} = \frac{\Delta t T_e^n + \Delta t f_2^n T_r^n + 2T_r^n \tau_R}{\Delta t + \Delta t f_2^n + 2\tau_R}, \tag{4.11}$$

$$\tilde{T}_e^n = \frac{\Delta t T_e^n + \Delta t f_2^n T_r^n + 2T_e^n \tau_R}{\Delta t + \Delta t f_2^n + 2\tau_R}, \tag{4.12}$$

$$T_e^{n+1} = \frac{\Delta t \tilde{T}_e^n + \Delta t f_1^n T_i^n + 2\tau_P^n \tilde{T}_e^n}{\Delta t + f_1^n \Delta t + 2\tau_P^n}, \tag{4.13}$$

$$T_i^{n+1} = \frac{\Delta t \tilde{T}_e^n + \Delta t f_1^n T_i^n + 2\tau_P^n T_i^n}{\Delta t + f_1^n \Delta t + 2\tau_P^n}. \tag{4.14}$$

and loop over all time steps.

We require an integration against which we can estimate errors of the foregoing method. We elected to use a semi-implicit non-operator scheme with a throttled timestep. After updating the nonlinear coefficients we use

$$\begin{aligned}
T_i^{n+1} &= \mathcal{C}\left[4\tau_P\tau_R T_r^n + \Delta t^2(T_e^n + f_1 T_i^n + f_2 T_r^n) + 2\Delta t((1+f_1)\tau_R T_r^n + \tau_P(T_e^n + f_2 T_r^n))\right], \\
T_e^{n+1} &= \mathcal{C}\left[4\tau_P\tau_R T_e^n + \Delta t^2(T_e^n + f_1 T_i^n + f_2 T_r^n) + 2\Delta t(\tau_R(T_e^n + f_1 T_i^n) + \tau_P(T_e^n + f_2 T_r^n))\right], \\
T_r^{n+1} &= \mathcal{C}\left[4\tau_P\tau_R T_i^n + \Delta t^2(T_e^n + f_1 T_i^n + f_2 T_r^n) + 2\Delta t((1+f_2)\tau_P T_i^n + \tau_R(T_e^n + f_1 T_i^n))\right],
\end{aligned}$$

with $\mathcal{C} = \Delta t^2(1 + f_1 + f_2) + 4\tau_P\tau_R + 2\Delta t(\tau_P + f_2\tau_P + \tau_R + f_1\tau_R)$, and loop over all steps n. Errors were calculated using

$$||\text{error}||_2 = \sqrt{\frac{\sum_n \max_j (T_j^n - \hat{T}_j^n)^2}{\sum_n \hat{T}_j^{n\,2}}},$$

where $\hat{T}^n$ is the temperature at time step n as derived from the throttled, non-operator split integration and $j \in \{e, i, r\}$.

4.3. **On the Unconditional Stability of Fully Implicit Operator Split Methods.**

Proof. Consider the test system

$$T' = \lambda T,$$

where λ is a constant. We can quickly write the above equation like

$$T' = (\lambda_1 + \lambda_2)T,$$

for an appropriate choice of λ_i. If we were to numerically integrate this problem with operator splitting (and utilizing an implicit Euler method with each step), we would use

$$\tilde{T} = (1 - \Delta t\lambda_1)^{-1} T^n \tag{4.15}$$

$$T^{n+1} = (1 - \Delta t\lambda_2)^{-1} \tilde{T} \tag{4.16}$$

We can eliminate $\tilde{T}$ to find that

$$T^{n+1} = (1 - \Delta t\lambda_2)^{-1}(1 - \Delta t\lambda_1)^{-1} T^n = \Phi(\Delta t\lambda_1, \Delta t\lambda_2) T^n,$$

where Φ is the amplification factor. For convergence we require

$$|\Phi(\Delta t\lambda_1, \Delta t\lambda_2)| < 1 \Rightarrow \left| \frac{1}{(1 - \Delta t\lambda_2)(1 - \Delta t\lambda_1)} \right| < 1.$$

If $\Delta t \to \frac{1}{\lambda_i}$ there will be stability issues. Because $\Delta t > 0$ there is no danger of instability so long as $\lambda < 0$ (as in a relaxation case). Stability issues only arise in the case of exponential growth ($\lambda > 0$). Because

$$|\Phi(\Delta t\lambda_1, \Delta t\lambda_2)| < 1, \quad \forall \Delta t > 0,$$

we say that Lie operator splitting with implicit Euler substeps is A-stable. Further

$$\lim_{\Delta t\lambda_i \to \infty} |\Phi(\Delta t\lambda_1, \Delta t\lambda_2)| = \lim_{\Delta t\lambda_i \to \infty} \left| \frac{1}{(1-\Delta t\lambda_2)(1-\Delta t\lambda_1)} \right| = 0,$$

so the method is also L-stable. □

References

Baldwin, C. Brown, P., Falgout, R., Graziani, R., Jones, J. 1999. Journal of Computational Physics. 154: 1-40.

Kozlov, R., Kvaernϕ, A., Owren, B. 2004. Journal of Computational Physics 195:576-593.

Masser, Thomas. 3T Timestep Controls Research Note. 2012. LA-UR Pending.

McClarren, R., Wöhlbier, J. 2011. Journal of Quantitative Spectroscopy and Radiative Transfer. 112: 119-130.

Knoll, D. A. Rider, W. J., Olson, G.L. 2001. Journal of Quantitative Spectroscopy and Radiative Transfer. 70: 25-36.

Wöhlbier, J. 2007. 3T model equations and operator split. LANL Research Note. LA-UR-07-4820.

New Algorithms for GPUs

(Bob Robey, mentor)

LA-UR-13-26528
Approved for public release; distribution is unlimited.

Title: A GPU Accelerated Discontinuous Galerkin Scheme for Advection

Author(s): Jibben, Zechariah J.

Intended for: Computational Physics Summer Workshop, 2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)
Report

Issued: 2013-08-19

A GPU Accelerated Discontinuous Galerkin Scheme for Advection

Zechariah J. Jibben [*†]

August 14, 2013

Abstract

This report describes the algorithms developed in OpenCL for solving a discontinuous Galerkin scheme to the linear advection equation used in interface transport. A sparse data structure was implemented to take full advantage of parallelism on the GPU, offering a final speedup ranging from 2-60x for a Nvidia Tesla C2050 vs. a 1.9GHz AMD Opteron 6168 running in serial. The algorithms take advantage of coalescence in global memory, as well as avoiding thread divergence.

[1]Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

[2]Supported by NSF grant NSF-CBET-1054272.

[*]Department of Mechanical and Aerospace Engineering, Arizona State University, Tempe, Arizona, 85287, E-mail: zjibben@asu.edu

[†]XCP-4 Methods and Algorithms Group, Los Alamos National Laboratory

1 Introduction

1.1 Motivation

A pressing problem in engineering is the modeling of fluid interactions involving an immiscible interface. For instance, inside a jet turbine where fuel is dispersed and atomized into air, the quality of the resulting mixture has a direct impact on the overall performance of the engine and pollutant production. Unfortunately, experiments are difficult or impossible to perform at operating conditions, simply because optical access is obstructed by engine structure and the "haze" of liquid droplets surrounds interesting structures. Furthermore, there is no known way of solving the nonlinear governing equations analytically. Therefore, numerical simulations become vital to the design process as well as to deepening our understanding of the physics, and having a high order solver becomes essential to accurately representing the material interface.

Here, the approach to this problem and benefit of utilizing GPUs to handle numerical algorithms is described. Here, sparsity becomes a noteworthy issue. Taking advantage of sparse data structures can be quite beneficial in CPU code. In a GPU implementation, however, it becomes crucial.

This paper begins by describing the problem in question, followed by a basic overview of the level set solution method and discontinuous Galerkin (DG) numerical method. The discussion of the scheme will involve noting the sparse nature of the system, followed by the GPU implementation (including some tricks that help in the process). Finally, results for the overall speedup are shown and discussed.

1.2 Governing Equations

Herrmann [3] gives a good overview for the governing equations of a fluid interaction involving immiscible interfaces. These are the Navier-Stokes' equations, along with a surface tension term $\boldsymbol{T}_\sigma$ that is nonzero only at the interface location $\boldsymbol{x}_f$.

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \left(\mu \left(\nabla \boldsymbol{u} + \nabla^{\mathrm{T}} \boldsymbol{u}\right)\right) + \boldsymbol{g} + \frac{1}{\rho}\boldsymbol{T}_\sigma \tag{1}$$

$$\boldsymbol{T}_\sigma(\boldsymbol{x}) = \sigma\kappa\delta\left(\boldsymbol{x} - \boldsymbol{x}_f\right)\hat{\boldsymbol{n}} \tag{2}$$

Here, $\boldsymbol{u}$ is the velocity, ρ is density, p is pressure, μ is dynamic viscosity, $\boldsymbol{g}$ is the gravitational body force, σ is the surface tension constant, κ is the local surface curvature, and $\hat{\boldsymbol{n}}$ is the local surface normal. As a result of this coupling from surface tension, an accurate method for tracking the phase interface location in such a way that allows us to also calculate the local curvature and normal at high order is vital. The level set method is selected to accomplish this goal.

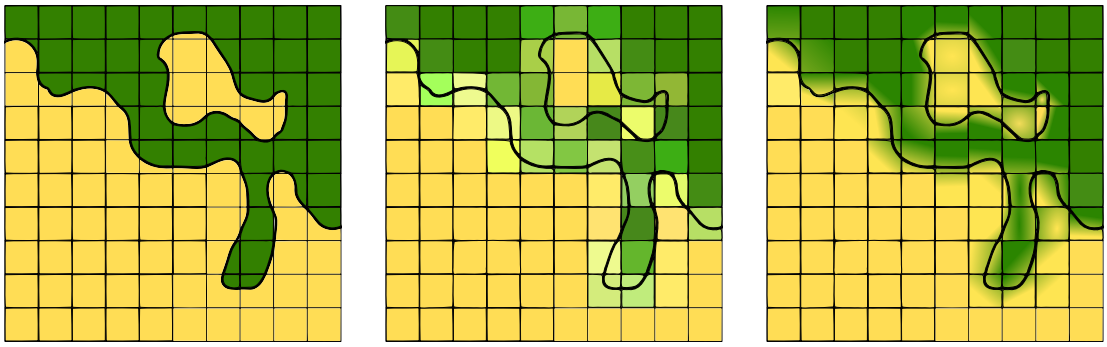
1.3 The Level Set Method

There are several approaches to interface tracking, volume of fluid methods (VOF) and level set methods being the most common. The VOF approach has the benefit of discretely conserving mass, while traditional level sets do not share this property. On the other hand, level sets have the benefit of high order accuracy and having the ability to compute high order normals and curvature. Recently, Olsson and Kreiss [7], Olsson et al. [8] developed a conservative level set method that treats the level set scalar as a conserved variable, greatly improving the mass conservation of the method.

The concept of level sets is to model the fluid interface, shown in Fig. 1, as the 0.5-isosurface of some scalar function $G(\boldsymbol{x}, t)$. Then, $G > 0.5$ on one side of the interface and $G < 0.5$ on the other. The 0.5-isosurface is then transported via the advection equation, which is found from the fact that the material derivative of $G\left(\boldsymbol{x}_f\right)$ is equal to zero. For incompressible flows, this can be written in conservative form as

$$\frac{\partial G}{\partial t} + \nabla \cdot (G\boldsymbol{u}) = 0. \tag{3}$$

Combining this approach with an arbitrary order discontinuous Galerkin method further improves the accuracy and mass conservation of level set methods.



(a) Fluid Interface (b) Finite Volume (c) Discontinuous Galerkin

Figure 1: Interface Discretization

2 The Discontinuous Galerkin Method

The numerical approach used is an arbitrary-order discontinuous Galerkin method, as described by Cockburn and Shu [1]. It can be thought of as a generalization of the finite volume method, which assigns average values of the solution variables to each cell. The discontinuous Galerkin method, on the other hand, allows sub-cell variation by performing a spectral decomposition of the solution variables in each cell. That is, we project $\boldsymbol{G}$ and $\boldsymbol{u}$ into the basis $\{b_i\}$ as

$$G_{\text{ic}} = \sum_{i=1}^{N_{\text{g}}} g_{i,\text{ic}} b_i, \quad \boldsymbol{u}_{\text{ic}} = \sum_{i=1}^{N_{\text{u}}} \hat{\boldsymbol{u}}_{i,\text{ic}} b_i, \tag{4}$$

where the series is truncated at N_g and N_u terms for $\boldsymbol{G}$ and $\boldsymbol{u}$, respectively (however, for this paper, we take $N_u = N_g$). In this sense, a finite volume method is equivalent to a discontinuous Galerkin method with $N_g = N_u = 1$. The normalized Legendre polynomial basis is selected for their orthonormality property, and they are constructed by performing Gram-Schmidt orthonormalization on the space of 3D monomials $x^\alpha y^\beta z^\gamma$. Then, for a maximum monomial degree k, we find $N_g = (k+1)^3$. It has been shown by LeSaint and Raviart [5] that this method can then formally achieve a $k+1$ convergence rate.

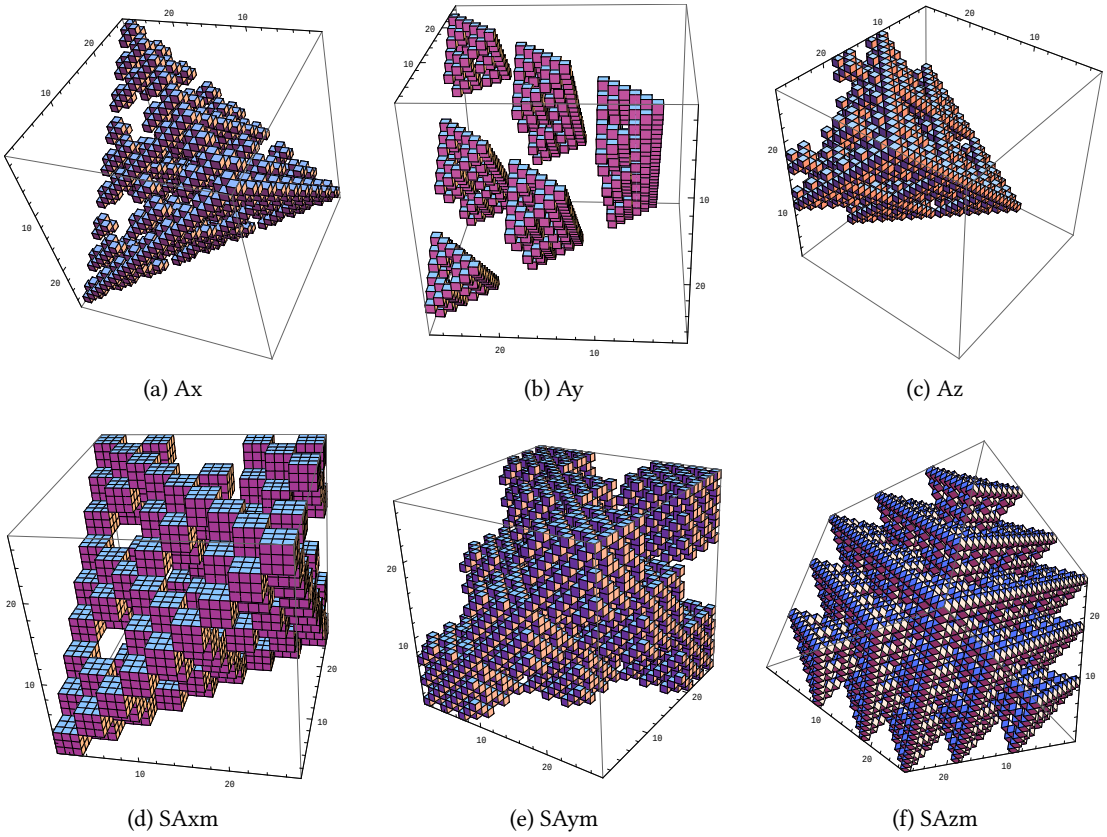
These expansions are then substituted into Eq. (3). By performing an inner product with b_n (integrate over the cell domain Ω), taking advantage of orthonormality, and using the divergence theorem, we arrive at a system of coupled ordinary differential equations describing the time evolution the coefficients g_n for all cells. A simple upwind flux is used to handle integration along cell interfaces, and a $k+1$ order Runge-Kutta (RK) time stepping mechanism is used.

$$\frac{\mathrm{d}g_{n,\text{ic}}}{\mathrm{d}t} = u^j_{k,\text{ic}} g_{i,\text{ic}} \int_\Omega b_k b_i \frac{\partial b_n}{\partial x_j} \,\mathrm{d}V + u^{j,\text{up}}_{k,\text{ic}} g^{\text{up}}_{i,\text{ic}} \int_{\partial\Omega} N_j b^{\text{up}}_k b^{\text{up}}_i b_n \,\mathrm{d}S \tag{5}$$

Note that the two integrals are entirely in terms of our basis functions and the cell domain. These can therefore be pre-computed analytically using symbolic software such as Mathematica, Maple, or SymPy, and stored in a 3D array for reference later. This avoids the use of quadrature, saving computation time. Furthermore, note that the orthogonality of the Legendre polynomial basis produces sparse arrays (see Table 1 and Fig. 2). For reference, the volume integrals are denoted Ax, Ay, and Az, and the "-" face surface integrals are denoted SAxm, SAym, and SAzm. Together, the high number of operations with comparatively few solution variables and the sparsity of the integral arrays make this method ideal for GPU computation.

Table 1: Matrix Fill Fraction

Polynomial Degree	2D Simulation		3D Simulation	
	Volume Integrals	Surface Integrals	Volume Integrals	Surface Integrals
1	12.5%	50.0%	6.25%	25.0%
2	10.6%	40.7%	4.30%	16.6%
3	10.1%	35.9%	3.63%	12.9%
4	9.68%	33.6%	3.25%	11.3%



(a) Ax (b) Ay (c) Az

(d) SAxm (e) SAym (f) SAzm

Figure 2: Sparsity illustration for 2^{nd} degree polynomials in 3D. Cubes are placed at array locations containing nonzero elements. Visualized by Mathematica.

It is highly beneficial to store these arrays in a manner that takes advantage of their sparse structure. A format similar to compressed row storage (CRS) [2] was chosen, allowing the integrals to be stored in 1D arrays along with three corresponding 1D arrays of ints giving nonzero element locations. By doing so, we limit the amount of data that must be sent to the GPU, and make parallelization on the GPU a simpler matter.

3 GPU Programming Model

Using OpenCL terminology, a GPU operates by executing a function called a *kernel* in parallel on a cluster of *work-items*, which are organized into *work-groups* with an associated memory space we call *tiles*. Each work-item then has a global id and local id, and each work-group has a group id. There are several nuances of this model to take advantage of, the most important of which is how workloads are managed. For instance, if work-items inside the same work-group are given drastically different workloads, the entire work-group must wait for the slowest member to complete its task before moving on to the next portion of the problem. As a result, it is highly advantageous to

assign uneven workloads to different work-groups, rather than within work-groups. Although the code is written with GPUs in mind, it can be effectively implemented on other architectures using the OpenCL model.

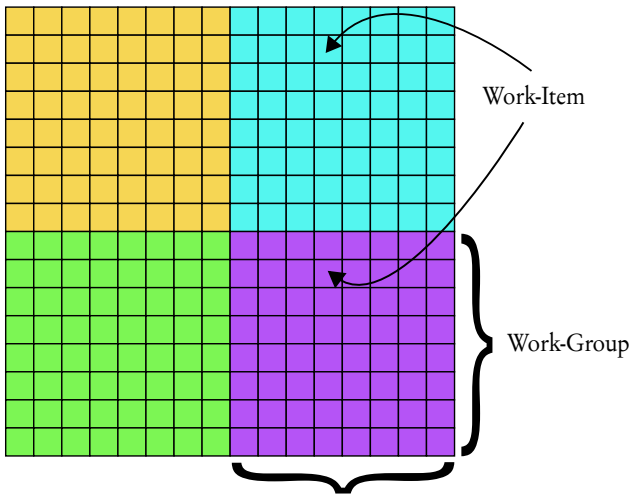


Figure 3: OpenCL Execution Model

To take advantage of this aspect of parallelism, and avoid thread divergence, Eq. (5) is solved by assigning a single $g_{n,\mathrm{ic}}$ to each tile. Then, work-items share the workload of tensor-vector multiplication and summation, avoiding uneven workload distributions between work-items (called thread divergence). This way, if an integral array has a largely zero row n, the entire work-group is given a lighter workload. This allows it to finish earlier and execute a new work-group rather than waiting for individual work-items to finish their work.

In Listing 1, Eq. (5) is considered a series of equations of the form $\Delta g_{n,\mathrm{ic}} \mathrel{+}= \sum_{k=1}^{N_u} u_{k,\mathrm{ic}} \sum_{i=1}^{N_g} g_{i,\mathrm{ic}} Z_{n,k,i}$. Each work-item has its own instance of the variable `my_dg`. Work-items then proceed to sum together a subset of the above equation, that is, products of elements of velocity `u`, level set scalar `g`, and the integral array, denoted `Z` for generalization in the code. Instead of looping over both k and i, we loop over a single integer l that corresponds to nonzero elements of the compressed array `Z`. Two arrays `Zi2` and `Zi3` give the values of k and i associated with l for each iteration. Finally, each work-group has its own value of $\Delta g_{n,\mathrm{ic}}$ to compute, with a unique combination of n and ic. Since each work-group only has one value of n, it only needs to loop through a subset of the integral array `Z`. As a result, it is necessary to pass in two integers `Znstart` and `Znend` that give the bounds of this subsection.

In order to take advantage of memory coalescence and evenly distribute the workload, and hence reduce runtime, the local group of work-items align their access to the array `Z` by their local id number. For example, the work-item with local id 7 will access the array element immediately after the work-item with local id 6 and immediately before the work-item with local id 8.

Listing 1: GPU Implementation

```
1   const uint tiX = get_local_id(0);   // get local work-item id
    const uint ntX = get_local_size(0); // get local work-group size
3
    // initialize summation variables
5   double my_dg=0.0;
    __local double partialsum[TILE_SIZE];
7
    for (uint l=Znstart+tiX; l<Znend; l+=ntX) {
9     uint k = Zi2[l]; uint i = Zi3[l];            // multiply by associated u and g
      my_dg += u[k]*g[i]*Z[l];
11  }
    partialsum[tiX] = term;                        // save private result to local array
13  my_dg = reduction_sum_within_tile(partialsum); // sum the partialsum elements
```

On the next iteration, l is updated with a step size equal to the number of work-items in the work-group. Finally, after each work-item has saved its result in an array stored in local memory, the elements of the array are summed together via a simple parallel reduction routine.

4 Tricks and Workarounds

Programming GPUs comes with the added challenge that OpenCL (and CUDA) currently do not support Fortran [9]. It is, however, possible for Fortran to call C functions. Since OpenCL readily supports host code written in C, functions in C can easily act as a staging area between Fortran and OpenCL. This is done by first giving C access to data allocated in Fortran, which is achieved by creating pointers to that data and sending them as arguments to a C function. Since Fortran natively sends pointers rather than the data itself, this is as simple as writing the first element of an array as the argument to a C function, which C then receives as a pointer to an array contiguous in memory.

Passing more complex data structures, such as arrays nested within arrays of derived data types, is more complicated, but still manageable. Because Fortran pads arrays in such a way that can be difficult to predict, it is simplest to send to C a pointer to the start of each array. This process can be made more compact by defining a derived data type in Fortran containing only a pointer, thereby allowing Fortran to generate an array of pointers (which is not natively available). A pointer to the first element of this array of pointers is then sent to C, which allows C to find the data associated with each variable within an array of derived data types.

Finally, OpenCL does not accept multidimensional arrays. To avoid bulky or obscure code, multidimensional arrays are sent to the GPU as 1D arrays (so long as they are contiguous in memory), and a macro is defined on the OpenCL side to simulate multidimensional behavior.

5 Results

The OpenCL algorithm for DG advection was executed on a Nvidia Tesla C2050 GPU (with a work-group size of 128) and compared to the original algorithm running in serial on a a 1.9GHz AMD Opteron 6186 CPU. Both algorithms take advantage of sparsity and are implemented on equidistant Cartesian meshes in unit sized domains. Verification of the method has been performed for the CPU algorithm previously [4], so the emphasis of this work is limited to compute times and assurance that the CPU and GPU give equivalent results (within 10^5 times machine epsilon at double precision). As such, the test problem is arbitrary. For robustness, the solution variable coefficients are randomized, and activating or deactivating terms of the equation can be used for debugging and additional assurance that each term is being evaluated correctly.

Table 2: Results for Compute Time of One RK-Step

Polynomial Degree	$1/\Delta x$	2D Simulation			3D Simulation		
		CPU time (s)	GPU time (s)	Speedup	CPU time (s)	GPU time (s)	Speedup
1	10	6.37e-4	2.12e-3	0.30x	2.25e-2	6.96e-3	3.23x
	20	2.52e-3	3.11e-3	0.81x	1.79e-1	3.99e-2	4.49x
	40	9.47e-3	6.28e-3	1.51x	6.18e-1	2.83e-1	2.18x
2	10	2.45e-3	2.45e-3	1.00x	3.24e-1	2.56e-2	12.7x
	20	9.63e-3	4.57e-3	2.11x	1.18	1.41e-1	8.37x
	40	3.85e-2	1.15e-2	3.35x	8.82	1.05	8.40x
3	10	8.81e-3	2.87e-3	3.07x	1.08	8.30e-2	13.0x
	20	3.38e-2	6.52e-3	5.18x	8.34	6.20e-1	13.5x
	40	1.34e-1	1.89e-2	7.09x	6.67e+1	4.78	14.0x
4	10	3.47e-2	4.34e-3	8.00x	1.15e+1	4.16e-1	27.6x
	20	1.38e-1	1.03e-2	13.4x	1.32e+2	3.03	43.6x
	40	3.92e-1	3.06e-2	12.8x	1.36e+3	2.40e+1	56.7x

These tests produce several interesting trends. First, low degree polynomials show little benefit from the GPU, and sometimes even slower runtimes. This is simply because of the parallelization scheme, where we delegate work

LA-UR-13-26533
Approved for public release; distribution is unlimited.

Title: Compact Hash Algorithms for Computational Meshes

Author(s): Tumblin, Rebecka
Ahrens, Peter
Hartse, Sara A.
Robey, Robert W.

Intended for: SIAM Journal of Scientific Computing

Issued: 2013-08-19

COMPACT HASH ALGORITHMS FOR COMPUTATIONAL MESHES

REBECKA TUMBLIN[†‡§], PETER AHRENS[‡¶], SARA HARTSE[‡‖], AND ROBERT W. ROBEY[‡]

Abstract. We employ compact hashing and the discrete properties of computational meshes to optimize spatial operations in scientific computing applications. As a model, we apply spatial hashing methods to the problem of determining neighbor elements in Adaptive Mesh Refinement (AMR) schemes. By applying memory savings techniques, we extend the perfect spatial hash algorithm to a compact hash by compressing the resulting sparse data structures. Using compact hashing and specific memory optimizations, we increase the range of problems that can benefit from our ideal $O(n)$ algorithms. The spatial hash methods are tested and compared across a variety of architectures on both a randomly generated sample mesh and an existing cell-based AMR shallow-water hydrodynamics scheme. We demonstrate consistent speed-up and increased performance across every device tested and explore the ubiquitous application of spatial hashing in scientific computing.

1. Introduction. The compact spatial hash algorithms we have developed extend the domain of hashing to include operations on finely resolved computational mesh structures. Robey, Nicholaeff, and Robey [16] demonstrated that mesh operations can exploit the properties of discretized data allowing for a hash-based approach to be implemented for spatial operations such as neighbor finding, sorting, remapping and table look-ups. Using their perfect spatial hash, they were able to achieve the ideal limit of $O(n)$ time. Computational meshes represent a subset of discretized data that we explore in this current work. With exascale computing on the horizon, memory operations and their consequent power usage are quickly becoming the dominant consideration for production codes in terms of speed-up and cost efficiency. We introduce memory efficiencies in the perfect spatial hash which allow hashing functions to be applied to a variety of physical applications by increasing the breadth of their performance enhancements to include a wide range of spatial structures.

Advances in high powered computing have led computational science to emerge as a new paradigm in science while acting as a crutch for both the theoretical and experimental fields. Most advanced production codes can take years to develop, and scientists often lag behind the most recent advances in hardware and algorithms due to the difficulty and time involved in integrating new techniques into pre-existing codes. As scientific computing becomes more cental to the scientific community, portability is quickly becoming an important consideration. We are able to show that compact hashes are robust, efficient, and easy to implement in highly resolved, complex physical applications while at the same time showing significant speed-up and increased performance on a variety of architectures.

As industry continues to proliferate parallel processors, adapting algorithms for fine-grained parallelism becomes necessary to fully utilize the capability of these machines. Hashing is an intrinsically parallel, non-comparison based search algorithm which requires no inter-thread communication. This allows performance portability across CPU and GPU architectures. Spatial hashing functions which take advantage of discretized data properties provide a new and innovative approach for adapting algorithms to more highly parallel architectures with broad application to computational science.

This paper is organized as follows. In Section 2, Background, we provide a synopsis of hashing and the advances that led to our research. In Section 3, Methodology, we describe

*Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

[†]Corresponding Author: email rtumblin@uoregon.edu

[‡]XCP-2 Eulerian Applications Group, Los Alamos National Laboratory

[§]University of Oregon

[¶]University of California, Berkeley

[‖]Brown University

the memory optimizations that enable the spatial compact hash. In Section 4, Performance Results, we explore performance on a randomly generated sample mesh and develop a cost model. In Section 5, Application to Scientific Computing, we demonstrate the application of compact spatial hashing to a cell-based AMR shallow-water hydrodynamics scheme, and we explore the performance enhancements achieved. In Section 6, Conclusions, we review the impacts these methods have in the broader scope of computation and where there is potential for further development.

2. Background. A hash table consists of an array of memory spaces, called buckets, where data may be stored. This data takes the form of values which are accessed by their associated keys. These keys are used to map their values to locations within a hash table. Every key maps to a unique integer, called a hash code, by means of a hash function. This unique integer is then converted to an index corresponding to a bucket in the table. We call this index a hash location. An entry (key-value pair) is inserted into a hash table, then the table is later queried for a key to get its associated value if it exists. More formally, a perfect (or 1-probe) hash function, h, is an injection (one-to-one) mapping such that:

$$h : U \to \{B_k\}$$

The set of keys, $\{S\}$, are members of the universe of possible values, $S_i \in U$and the buckets $\{B_k\}$ comprise the hash table with a size $|\{B_k\}|$. In addition, for $x, y \in \{S\}$ such that $x \neq y$, it must be true that $h(x) \neq h(y)$. This last guarantee is that no collisions may occur.

A perfect hash table is designed for data whose range and pattern of hash codes are known, so that creating a hash table large enough to accommodate all the possible keys is feasible. A compact hash uses a compression function to reduce the range of hash locations. If it cannot be guaranteed that the hash codes will be unique, we must make plans to overcome the situation in which two different keys have made their way to the same bucket. This situation is referred to as a collision, and compact hashes are designed to handle collisions.

One of the earliest collision handling methods, chaining, developed by Williams [17], allows multiple keys to exist in one bucket through the use of singly-linked lists (chains of pointers, leading to this method's name). Another of the early approaches is to use a separate hash table to store colliding entries, called double hashing. An interesting approach to using the existing memory for the hash table was developed by Peterson [15] and called open addressing. Peterson proposed, in the event of a colliding entry, to search to the right of its original location in a predetermined, repeatable sequence for an empty bucket in which to insert the entry. Regardless of how collisions are resolved, they are always more expensive to handle than simply inserting an entry into an empty bucket. The minimization of collisions is a key aspect of compression function design. Thus, the compression function in a compact hash table is designed to randomize the hash locations to reduce the number of collision due to patterns in the data. The table size then has no relation to the range of possible keys, but rather the number of entries to be inserted. The number of entries divided by the number of buckets in a hash table is referred to as the load factor.

Advances have been made to both perfect hashing and compact hashing since their creation. Czech, Havas and Majewski[5] present a thorough treatment of enhancements to perfect hashing in their monograph. Most of the monograph deals with static data sets where it is worth a lot of compute time to come up with a perfect hash function. But the last section of the monograph discusses dynamic data sets and also has a very short section on parallelization efforts for these methods. For the compact hashing domain, Munro and Celis [14] present some advanced reordering techniques such as Brent's method. Brent[**?**] proposed the smart placement of keys with the plan to reduce the number of probes needed for queries.

These reordering techniques are based on the idea that when there is a collision, there is a choice of which key-value pair to move down the probe sequence.

The recent interest in parallel programming has led to the realization that hash tables are easy to parallelize. The nature of the perfect hash function mostly requires no inter-thread communication, allowing for fine-grained parallelism to be exploited. Each insertion or query can function alongside one another since each bucket holds its own unique key. This is unlike a k-D tree method where searching operations are dependent on the previous iteration. A k-D tree bisects the data, and searches to the left and right and then continues unitl the queried data is found. This algorithm can be achieved in $O(n \log n)$ time. In a hash based approaches, we directly query the memory location of the desired value. Hash based approaches execute in expected $O(n)$ time.

Implementations of compact hashing with its collisions also introduces some dependencies on adjacent threads though to a lesser degree than a tree-based method. In highly parallel environments, the collision handling can generate performance and correctness issues. These can be resolved with a locking mechanism, but more recent work by Gao, Groote, and Hesselink [7] developed a lock-free and nearly wait-free implementation of open-address hashing. An interesting aspect of their work is the lengthy proof of correctness they felt necessary to undertake for their hash implementation.

Alcantara et al. [2, 1] saw that advancing technology allowed for efficient implementation of hashing methods on GPU's. He developed a set of parallel implementations for open addressing, chaining, and cuckoo hashing on GPU's. He then examined their performance based on memory usage, table construction speed, and retrieval efficiency. Alcantara showed that tables with millions of elements can be effectively implemented by using the new atomic operations on the GPU's. He also noted that each application requires different considerations for the inclusion of specialized features when developing the hashing method. Ultimately, a balance between memory usage, table construction speed and retrieval efficiency must be tailored to each specific application.

3. Methodology. We now wish to apply the attributes of hashing to operations involving data associated with computational meshes and the spatial layout of a grid. The hash-based spatial method treats a mesh's cells as discretized data and uses a hash function to map this data to a hash table. For our definition of discretized data we mean a collection of information that is mapped onto a matric space$(A,\, d_A)$ with the properties defined by Robey, Nicholaeff, and Robey [16]. Spatial hashing is characterized as mapping a metric space $(A,\, d_A)$ containing differential discretized data to the powerset of a finite cover $\{B_k\}$ of a new metric space $(B,\, d_B)$ whose elements form a hash table. Note that the use of a metric space endows a distance attribute to the set, distinguishing this concept as spatial hashing, in contrast with the general hashing concepts introduced earlier. We define this mapping as a hash function, h:

$$h : A \to \mathcal{P}\left(\{B_k\}\right) \quad (A_i \not\mapsto \emptyset \; \forall i)$$

where $\{B_k\}$ represents a bucket in the hash table. The hash table size, or finite cover, can then be defined as,

$$|\{B_k\}| = \frac{m(A)}{|\Delta_{\min}|}$$

where $|\{B_k\}|$ is the set of buckets, $m(A)$ is the range of the original data being hashed, and $|\Delta_{\min}|$ is some minimal integer size of the dataset. Mapping to a powerset allows the possibility of mapping elements of $\{A\}$ to multiple locations in the hash table. If we know the range our data encompasses, the maximum and minimum values it can take, $m(A)$, and we

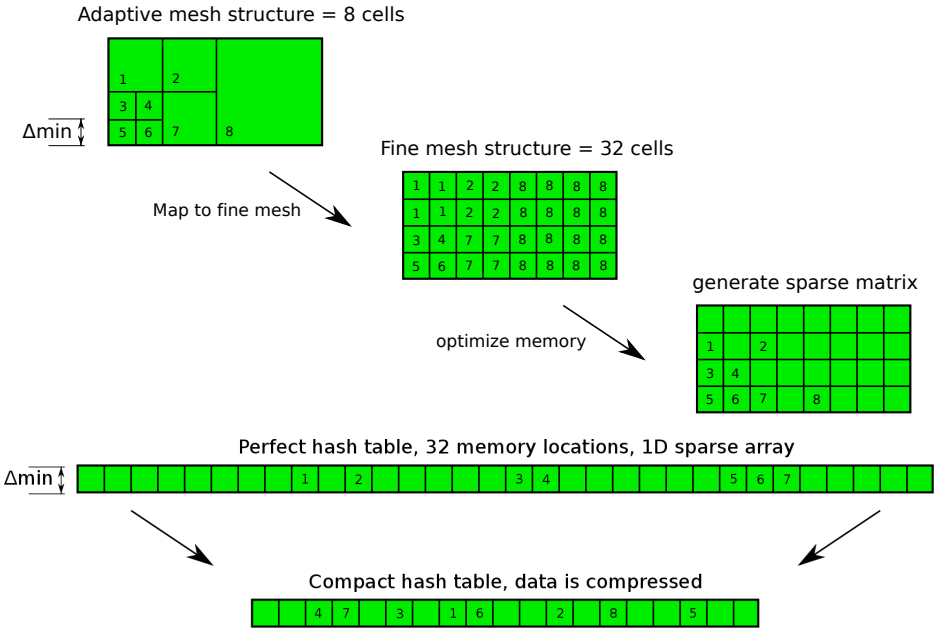


FIGURE 3.1. *This concept diagram depicts the steps of hashing spatial data in AMR.*

know the minimum difference between values, $\triangle_{min}$, this allows us to generate a perfect hash table containing the minimum number of buckets for an injective mapping. By choosing $\triangle_{min}$ to be the minimal distance between cell centers on a computational mesh, we can guarantee a bucket for the data contained in the computational mesh. Extending this formalism to compact hashing involves compressing the range of the new metric space (B, d_B) so that there is no longer a one-to-one correspondence between the original elements of the metric space and locations in the hash table. We explore specific methods for ahcieving this while at the same time preserving the relevant data in the original set in Section 3.2.

To help elucidate the multiple layers of abstraction, we wish to provide a short conceptual run-through of spatial hashing in the context of AMR datasets. However, we do not want to limit the scope of spatial hashing to this one specific application. We will be using cell locations in our concept diagram, but physical data such as density fields could be used instead. Our approach involves mapping the AMR cell locations to the fine mesh structure as shown in Figure 3.1. By exploiting the meshes inherent properties, we can reduce the number of writes which uniquely determine cell locations; these methods will be explored in Section 3.2. Reducing the writes generates a sparse matrix which contains the spatial layout of the computational mesh.

Once we have reduced the number of writes, we map the mesh structure to a sparse, one-dimensional perfect hash table. The sparsity of the mesh makes it a viable candidate for compression. We can compress the data into a compact hash table using a compression function which randomizes the data to reduce the number of collisions. Later in the process, we will have to deal with the collisions. The terminology for perfect and compact spatial hashing are shown in Figure 3.2.

We rely on the spatial layout of the computational mesh to generate a unique key associated with each value. The idea of a unique key is explored in depth in an earlier paper by

Terminology:

	compressibility	sparsity	load factor
perfect hash	32/8 = 4	8/32 = .25	8/32 = .25
compact hash	24/8 = 3	8/24 = .33	24/32 = .75

FIGURE 3.2. *This table defines compressibility, sparsity and load factor based on the examples in the previous figure.*

Robey, Nicholaeff, and Robey [16]. This key for a cell-based AMR grid is the spatial location of the cell at the finest level of the mesh. This directly corresponds to the hash bucket index for the perfect hash. The bucket index is simply the unique key generated from the $(i, j, level)$ location and becomes just a simple array assignment or retrieval as shown in Listing 1. In our current extension to compact hashes, we continue to exploit the unique key.

LISTING 1. *Coding for a Perfect Hash*

```
1 //* insert operation
2 hash[inputKey] = inputValue;
3
4 //* query operation
5 return(hash[inputKey]);
```

3.1. The Neighbor Finding Problem. In this paper, we only address the neighbor finding operation for a cell-based AMR application. This spatial operation is important to the performance and scalability of these codes. Optimizing each spatial mesh operation can only be done by exploiting the details of the application. There are two unique steps to this optimization. First, we reduce the number of writes to the minimum needed to support the spatial queries (Section 3.2). Second, we compress the data in the fine, regular grid structure of the mesh into a smaller compact hash (Section 3.3). We will exploit some of the application rules in this process. The most important rule is that refinement jumps can only be a factor of two at each interface. This means that each adjacent cell can either be a factor of two finer, the same size, or a factor of two larger. The second rule for this use case requires that the hash table is temporary, it is constructed and then used only once per iteration. If it is fast enough to be used in this scenario, then memory can be freed for other needs in the computation. But there is still the flexibility to keep the hash table and make modifications to the table every iteration, saving much of the table initialization and setup time.

Other operations such as remapping and table look-up can also be optimized in a similar manner, but the details of the spatial operation and the rules of the application must be considered in the process. This same methodology can be applied to other applications such as cell-based AMR with different refinement rules, patch-based AMR and unstructured general polyhedral grids.

3.2. Memory Write Optimizations. The nature of the neighbor finding problem and the structure of the mesh determine whether the insertions and queries performed in the perfect spatial hash are actually necessary. We can exploit aspects of the neighbor finding problem to reduce the number of insertions necessary in the perfect hash table, as illustrated in Figure 3.3. Optimization 1 only writes to the outermost cells because the interior cells are never accessed and, thus, irrelevant for determining a neighbor cell. Interior elements are given a null value, represented by a sentinel value. Optimization 2 takes advantage of the fact that the level of refinement between cells can only differ by one symmetric bisection and

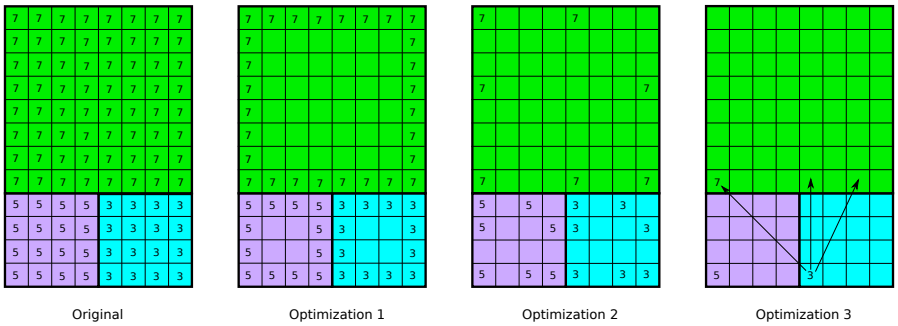


FIGURE 3.3. *Memory write optimizations introduce sparsity in the hash table.*

reduces the writes to only seven outer cells. Optimization 3 extends this idea, reducing the process to only one write with three reads, each checking for a finer, same size, or coarser cell. All of our optimizations involve only mapping some elements of the original dataset, $(A,\, d_A)$, to the hash table $(B,\, d_B)$ while preserving necessary information. This allows us to compress the number of buckets in the hash table without affecting the relevant data containing the cell locations on the adapted mesh.

Reducing the insertions and queries speeds up the execution time of the perfect hash and, as it reduces the number of necessary buckets, introduces sparsity into the problem. Sparsity is related to memory compressibility of the mesh, which we define as the ratio of the maximum number of cells on the finest level of the mesh to the number of cells in the current timestep of the simulation (See Figure 3.2). Also, reducing the number of insertions helps with thread divergence issues on the GPU by decreasing the difference between number of insertions and queries between adjacent threads. Thread divergence occurs when threads in a branching operation wait for the slowest thread to finish its calculation before advancing to the next instruction.

Now, with the reduction in insertions, the compact hash is viable because it is able to take advantage of this memory compression and make smaller hash tables, offsetting the much more costly insertions and allowing for the possibility of much larger problems.

The specifications of the grid we are examining, as well as the requirements of the neighbor finding problem, allow for these particular memory optimizations and the introduction of sparsity. In order to implement the compact hash for other spatial operations, it will be necessary to find comparable optimizations.

3.3. The Compact Hash. When making design decisions for the compact hash on the CPU and GPU, priority was given towards optimizing the GPU implementation, while making the two implementations as similar as possible to avoid code repetition. One example would be our choice of open addressing. Open addressing works incredibly well on the GPUs, but for the CPU, we could have used chaining in place of open addressing. Thus, some decisions made when optimizing the GPU governed the CPU algorithms.

In a perfect hash table, we don't have to handle collisions because each key's hash code is its hash location. When implementing a compact hash table, not only must we handle collisions, we must reduce their frequency of occurrence. By randomizing the hash codes before reducing their range to fit in the hash table, patterns in the input data that would normally lead to collisions can be avoided. Mitzenmacher and Vadhan [12] showed that a 2-universal compression function is sufficient to be competitive with more complex methods

even though it is simple to implement and cost-effective to execute. The definition of what it means to be 2-universal is given by Carter and Wegman [4], and we advise the reader to refer to his work for a more complete understanding of this family of functions. Our compression function is taken from the first example of such a function given by Carter and Wegman:

$$hashLocation = ((a * hashCode + c)\%m)\%tableSize$$

where a, c and m are constants and the modulo ($\%$) operator is used to generate randomness and keep the result in the the range of the table. The constants a and c must satisfy certain constraints to achieve good performance, the same constraints necessary to guarantee a full period of the similar-looking linear congruential pseudorandom number generator [8, 13]. The factors c and m should be relatively prime, $a - 1$ should be divisible by all prime factors of m, and $a-1$ should be a multiple of 4 if m is a multiple of 4. Our initial implemention uses simple random numbers, but for the library under development, we used accepted constants for linear congruential generators ($a = 2147483629$, $c = 2147483587$, m is the largest 32-bit prime). We plan on randomly generating numbers that satisfy the above contraints.

Although they can be reduced, collisions cannot be avoided entirely. To handle collisions, a quadratic probing open addressing hash table with the linear congruential compression function was chosen due to the superior construction and retrieval speeds for large data sizes on the GPU as shown by Alcantara [1]. When inserting or querying within a hash table with open addressing, the goal is the same: to find a bucket containing the given key or to find an empty bucket where the key can be placed. If we are inserting into the table, finding an identical key allows us to overwrite the entry there. Finding an empty location would allow us to simply insert the entry there. When querying the table for a key, finding the key means that we can return its associated value. Finding an empty bucket means that the key does not exist. The search for such a bucket is the same whether we are inserting into or querying the table. We obtain a hash location from the compression function and hash code (we get the hashcode from the hash function and key). We first check the bucket at this location to see if it is null or contains a matching key. If not, we check buckets at locations in the sequence according to:

$$hashLocation_i = (hashLocation_0 + P(i))$$

where P is some predetermined probe sequence.

In a linear probing open addressing hash table, the probe sequence is:

$$P_{linear}(i) = B * i$$

It has been shown that 1 is a good value for B [8]. Linear probing does have some issues however, as long clusters of keys are created that must be traversed iteratively to find empty locations or keys. This effect is called primary clustering, and can be remedied through the use of a quadratic probe sequence as developed by Mauer [11]. A quadratic probe sequence is:

$$P_{quadratic}(i) = A * i^2 + B * i$$

It has been shown that good values for A and B are 1 and 0 [8]. Quadratic probing solves the problem of primary clustering, but suffers from another effect, secondary clustering. This is the issue of chains forming when multiple keys are mapped to the same initial spot, and must probe the same sequence to find an empty spot to insert. It is known that the first $tableSize/2$ locations probed in a quadratic probe sequence will be unique if the tablesize is prime [3]. We

need a prime number, and since there are simple tests for the primality of proth numbers, we generate a proth prime close to the desired table size. Proth primes occur with a reasonable frequency. For a compact hash using open addressing with a quadratic jump, the operations can be expressed in a single yet complex loop as shown in the pseudocode in Listing 2 and C code for the CPU in Listing 3.

LISTING 2. *Pseudocode for Compact Hash using Quadratic Probing*

```
//* insert operation
set hashLocation to ((inputKey * a + c) % m) % tableSize
do{
    set insertLocation to next location in quadratic probe sequence
}while insertLocation isn't a valid spot for insertion and we haven't tried too many
     times
if insertLocation is a valid spot for insertion{
    set the key at insertLocation to inputKey
    set the value at insertLocation to inputValue
}

//* query operation
set hashLocation to ((inputKey * a + c) % m) % tableSize
do{
    set queryLocation to next location in quadratic probe sequence
}while the bucket at queryLocation isn't empty and doesn't match our inputKey and we
     haven't tried too many times
if the bucket at queryLocation is empty{
    let user know query is unsuccessful
}else if the keys match{
    return the value at queryLocation
}
```

LISTING 3. *Compact Hash Coding for the CPU using Quadratic Probing*

```
//* insert operation
iteration = 0;
hashLocation = ((inputKey * a + c) % m) % tableSize;
for (insertLocation = hashLocation;
    hash[2*insertLocation] != -1 && hash[2*insertLocation] != inputKey && iteration <
         maxTries;
    insertLocation = (hashLocation + iteration * iteration) % tableSize) { iteration++;}
if(iteration < maxTries){
    hash[2*insertLocation] = inputKey;
    hash[2*insertLocation+1] = inputValue;
}

//* query operation
iteration = 0;
hashLocation = ((inputKey * a + c) % m) % tableSize;
for (queryLocation = hashLocation;
    hash[2*queryLocation] != -1 && hash[2*queryLocation] != inputKey && iteration <
         maxTries;
    queryLocation = (queryLocation + iteration * iteration) % tableSize) { iteration++;}
if (hash[2*queryLocation] == inputKey){
    return &hash[2*queryLocation+1];
}else{
    return NULL;
}
```

Hashing in parallel is very similar to hashing serially. Every thread is assigned an entry to insert, or a key to query, and all threads probe the same hash table in global memory in parallel for the appropriate locations. This model is particularly suited towards GPU computing, as GPUs have a fair amount of relatively fast global memory on the device. The parallel insertion algorithm differs from the serial one in one important way. When checking to see if a bucket is empty and, if it is, inserting a key in it, we must impose mutual exclusion on the bucket's key field and its empty or full field. This means that for the duration of the above set of operations, only one thread may access these fields at a time. If a sentinel key value is chosen to mean that a bucket is empty, this mutual exclusion can be performed with an atomic check

and exchange operation. For an operation or set of operations to be considered atomic, it must be performed without interruption. While not all atomic operations are implemented using a form of mutual exclusion, they must guarantee the behavior. If the insertion thread did not use one of these techniques during its execution, it would be possible for two threads to check if a bucket was empty, and both write their (potentially different) keys into the same memory address. This creates a race condition, where the key that gets written depends on which thread reaches the memory address first. Obviously, this must be avoided because the other thread will, in this situation, fail to write it's entry into the table at all. The pseudocode for a parallel insert in shown in Listing 4. In OpenCL, there are atomic operations defined for 32-bit and 64-bit integers, and if a sentinel value is used to mean that buckets are empty, then mutual exclusion doesn't need to be imposed by more explicit means. An example of OpenCL code with an atomic operation is shown in Listing 5.

LISTING 4. *Psuedocode for the Parallel Insert Operation*

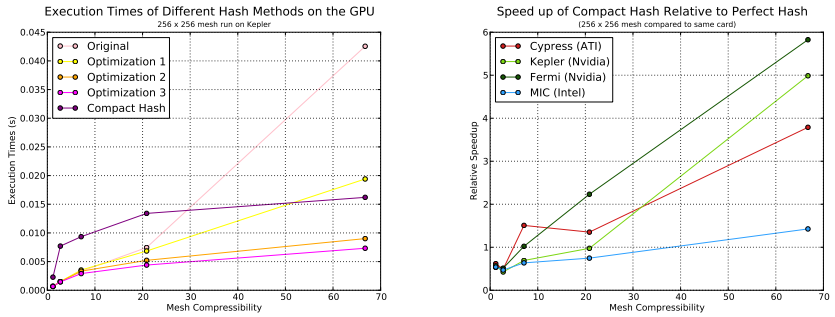
```
set hashLocation to ((inputKey * a + c) % m) % tableSize
while we haven't tried too many times{
    set insertLocation to next location in quadratic probe sequence
    aquire lock on the key memory at insertLocation
    if insertLocation is a valid spot for insertion{
        set the key at insertLocation to inputKey
        unlock the key memory at insertLocation
        set the value at insertLocation to inputValue
        break
    }
    unlock the key memory at insertLocation
}
```

LISTING 5. *Compact Hash Coding for Parallel Insert Operation*

```
iteration = 0;
hashLocation = ((inputKey * a + c) % m) % tableSize;
oldKey = atomic_cmpxchg(&hash[2*insertLocation], -1, inputKey);
for (insertLocation = hashLocation;
    oldKey != -1 && oldKey != inputKey && iteration < maxTries;
    insertLocation = (hashLocation + iteration * iteration) % tableSize) {
    iteration++;
    oldKey = atomic_cmpxchg(&hash[2*insertLocation], -1, inputKey);
}
if(iteration < maxTries){
    hash[2*insertLocation] = inputKey;
    hash[2*insertLocation+1] = inputValue;
}
```

3.4. Delivering a GPU Library. A library for hashing both on the CPU and parallel devices was developed to better create an abstraction barrier between neighbor calculation and hashing, to improve code modularity and reuse in our organization, and to encourage others to adopt and explore hashing techniques. The library was created in C for use by Fortran C, and C++. OpenCL was chosen as a parallel computing language for it's portability across different GPU devices.

The software coding techniques for delivering an OpenCL library are very new. OpenCL compiles at runtime on the currently selected hardware, yielding strong portability. A string containing the code is compiled for that hardware during execution. But for portability, there are two situations that must be handled. The first situation is the kernel source code should be bundled with the library so that the library and source code cannot get separated. Conventional techniques involve storing the CL code in a file and reading that file when the string is needed. We developed a string encapsulation for the source code into the hash library. The run-time compile of the OpenCL code now uses a string embedded into the library instead of a separate file. Thus users no longer need to specify the path to the OpenCL file. Some of the functions in the library are subroutines rather than kernels. These must be present in the



a. b.

FIGURE 4.1. *a) Memory optimizations improve performance. b) The compact hash method shows speed-ups on different devices.*

application's OpenCL source at run-time. To handle this case, the source code is embedded in a string and the application retrieves it with a get source command, prepends it to their application OpenCL code and then compiles it.

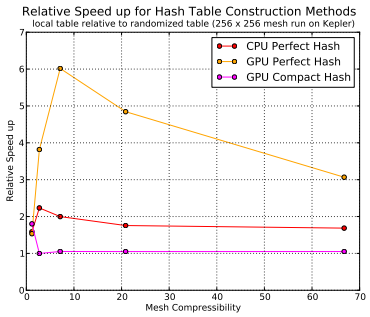
In OpenCL, function pointers and arrays of memory pointers are not allowed, rendering many object-oriented design concepts obsolete. To work-around this, an object-based approach using macros was taken for the design of the hash library. Macros allowed us to hash values of a generic type by imitating polymorphism and inheritance. The heavy use of macros also allowed us to code key sections for both OpenCL and C, as the respective syntax does not vary too much. To allow us better error detection, the macros were expanded into separate files so that the compiler could use the correct line numbers. Another advantage of using macros in OpenCL is that code passed to the hardware is already in string format and doesn't need a separate file. This in effect embeds the OpenCL source code and in the process solves the problems discussed above.

We considered many use cases in developing a versatile OpenCL library. The data to be operated on may be on the host, on the device, or some combination of the two. The library must be capable of handling all of these cases. In order to do so, we automatically transfer data if necessary to support each situation.

We are including the version of the hash library used in this paper as supplemental material. This will improve the reproducibility of this research. It will be released with the Apache license which allows free use, but requires attribution so that the impact of this work may be ascertained.

4. Performance Results. After developing the compact hash algorithm utilizing the rules of our AMR method to reduce memory usage and generate sparse data structures, we implement our methods on a randomly generated sample mesh to test their performance capabilities. We test the different memory optimizations we developed, as well as, explore performance variation which occurs due to spatial locality within the mesh structure. Our methods were developed to be portable across a variety of architectures making it necessary to consider the impact hardware will have on performance and protability. After we obtain performance results, we develop an analytic model and compare to our numerical results.

4.1. Standalone neighbor testbed. Here we investigate the performance of our spatial compact hash to determine neighbor elements on a 256 x 256 coarse grid refined up to five levels. The results are shown relative to the mesh compressibility of the given problem.
Figure 4.1a shows the timings of each successive optimization as applied to the perfect hash,



a.

FIGURE 4.2. *a) Randomizing the table contruction shows the effect of cache on algorithm performance.*

as well as, the timing of the compact hash using optimization 3. Each of the optimizations adds improvement from the original perfect hash, with an especially significant difference between the original method and optimization 1. Initially, the compact hash surpasses even the original perfect hash indicating that for smaller scale problems with fewer refined cells, the perfect hash is preferable. With increasing refinement, the performance of the perfect hash using the original method and optimization 1 begins to decline and the compact hash, despite its initial offset, is faster. For the problem sets tested, the last two memory optimizations remain faster, but the trend of scalability with mesh compressibility is a good sign for the utility of the compact hashing for large scale, compressible problems.

Figure 4.1b shows the compact hash tested across a variety of GPU architectures. These include Nvidia's M2090 (Fermi) and K20Xm (Kepler) cards (released in 2011 and 2012 respectively and both members of Nvidia's General Purpose GPU Tesla family), ATI's Firepro V7800 (Cypress) and Intel's Many Integrated Core (MIC). The timings are presented as a speed-up relative to the perfect hash on each card. As previously observed, the compact hash does not improve upon the perfect until a compressibility threshold is reached, this threshold varies between devices. Despite being a CPU device, the MIC shows performance on scale with the GPUs. Its lower numbers could be attributed to the difference in the multiprocessor architecture and the fact that we did not optimize the workgroup-size specifically for the MIC. The MIC software is relatively new, and as we get more experience the performance should improve. Apart from this, these results are a strong indication of performance portability which, when combined with the code portability offered by OpenCL, demonstrates that the speed-ups are relatively easy to attain across architectures. These consistent performance enhancements are most likely a function of improvements at the algorithmic level which expose and utilize parallelism inherent to the problem, as opposed to custom programming for the individual devices. OpenCL is designed to be highly portable, allowing us to write routines once that can then be compiled and run on any OpenCL-compliant hardware, including multiple types of devices simultaneously.

In addition to devices, another variable to consider is cache storage and data locality. A test case for a random problem involves generating the sample mesh's refinement randomly and as well as randomizing keys in the compact hash table during the compression stage. This means that the program does not take advantage of any spatial locality. In a physical application the mesh will not be random, there will be patterns inherent in the mesh structure, and there may be some benefit to taking advantage of the cache.

We experimented with turning off randomization in generation of the hash table which adds correlation between the spatial data and the hash table. As seen in Figure 4.2a, a spatially

organized hash table offers speed-up in the perfect hash methods, especially on the GPU, but does not make a significant difference for the compact hash. This is understandable since the compression scheme results in scattering. The question of utilizing cache efficiency on the GPU has to be countered against the necessity for sparsity. Taking advantage of locality with the perfect hash offers speed-up on the GPU, but introducing compressibility in the compact hash allows it to scale to larger problems.

4.2. Cost Model. The cost model for the neighbor finding calculation is composed of an initialization of the hash table to the sentinel value, the insertion of all of the cell data, and the query for four neighbors for each cell. The cost model is complicated by the probabilities of collisions and the number of queries needed to find either a fine neighbor, same size neighbor or coarser neighbor.

Perfect Hash Cost (C_{PH}):

$$C_{PH} = HashTableSize * C_{Init} + N_{Cells} * C_{Write} + 4 * NQueries_{ave} * N_{Cells} * C_{Read}$$

where $1 < NQueries_{ave} < 3$ are the average number of queries to find the neighboring cell with tries at the finer level, same level and finally a coarser level. $NQueries_{ave}$ should be close to 2 for a typical AMR mesh.

Compact Hash Cost with generic probing (C_{CH}):

$$\begin{aligned} C_{CH} = & \frac{N_{Cells}}{LF} * C_{Init} \\ & + N_{Cells} * ((C_{Probe} + C_{Atomic}) * SuccProbeSeq_{ave} + C_{Write2Words}) \\ & + N_{Cells} * (C_{Probe} * (SuccProbeSeq_{ave} + \\ & \quad (NQueries_{ave} - 1) * UnsuccProbeSeq_{ave}) + C_{Read2Words}) \end{aligned}$$

where $LoadFactor(LF) = N_{Cells}/HashTableSize$. Note that the compact hash has to write both the key and the value to aid in the collision handling, whereas the perfect hash only has to write the value. These are distinguished by the $C_{Write2Words}$ and $C_{Read2Words}$ in the compact hash and the C_{Write} and C_{Read} in the perfect hash. Though twice as much information is being written or read, cache effects make the cost of the additional reads and writes less than a factor of two.

The compact hash cost with linear probing (C_{CHL}) and compact hash cost with quadratic probing (C_{CHQ}) are obtained by substituting $LinProbeSeq_{ave}$ or $QuadProbeSeq_{ave}$ for $ProbeSeq_{ave}$ in the above equation. Note also that the initial insertion will never have a collision, but by the final insertion, the collision probability is the load factor of the table. Thus the collision cost must be integrated over the insertions starting at 0 and ending at the final load factor. There is a closed form solution for the linear probes (from Knuth[8]), but the quadratic probe sequence contains an integral in the equation (from Bell[3]).

Average successful linear probe sequence, $SuccLinProbeSeq_{ave}$

$$\frac{1}{2}\left(1 + \frac{1}{1 - LF}\right)$$

Average unsuccessful linear probe sequence, $UnsuccLinProbeSeq_{ave}$

$$\frac{1}{2}\left(1+\frac{1}{(1-LF)^2}\right)$$

Average successful quadratic probe sequence, $SuccQuadProbeSeq_{ave}$

$$\frac{1}{LF} * \int_{x=0}^{LF} \left\{ -\left(\frac{1}{LF}\right) ln(1-LF)(LF-(1-e^{(-LF)}))dx \right\}$$

Average unsuccessful quadratic probe sequence, $UnsuccQuadProbeSeq_{ave}$

$$\frac{1}{LF} * \int_{x=0}^{LF} \left\{ -\left(\frac{1}{LF}\right) ln(1-LF)(LF-(1-e^{(-LF)}))dx \right\} - \frac{ln(1-LF)}{LF}$$

To get a sense of the magnitude of the collision frequency and cost, some examples are helpful. In a hash table with chaining, a load factor of 0.5 will result in an average of 1.25 buckets that are checked per search. In a hash table using open addressing for collision resolution, a 0.3 load factor will traverse 1.21 buckets per search on average [8]. The lower load factor for the open addressing is due to collisions being resolved within the table memory space rather than in an external linked list. The additional cost for the collisions is not simply the average buckets touched which would only be 20-25% more. The key and value must be stored so that the owner of the bucket can be determined, whereas without the possibility of collision, only the value must be stored. Offsetting this is the reduced cost of the initialization of the smaller hash table.

5. Application to Scientific Computing. Many physical applications in scientific computing rely on acquiring numerical solutions to partial differential equations (PDEs) on a discrete grid structure that approximates a continuous space. The size of each grid element is directly proportional to the numerical error associated with algebraically approximating solutions to PDE's. Often, physical applications estimate solutions across steep shocks or gradients where the physics changes rapidly in time. Resolving discontinuities on a discrete mesh requires finely spaced grid elements, and usually, the discontinuities that arise only cover a sparse region of the grid. It is therefore inefficient to finely resolve the entire mesh. AMR allows finer resolution over regions of interest while keeping the computational costs down over regions where low resolution is suitable (see Figure 5.1). We have shown that the compact hash offers a robust, efficient, and easy to implement method for neighbor determination on a randomly generated sample mesh. We now wish to demonstrate that our methods are viable to implement in a working scientific application where complications such as patterns in the spatial data arise. Our other goal in this section is to show that the compact hash scales to large, finely resolved meshes.

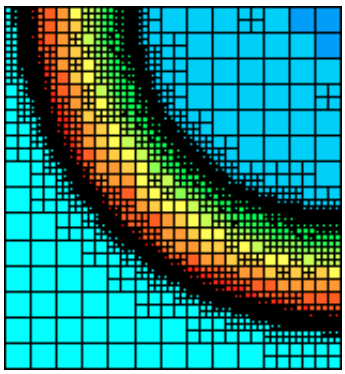


FIGURE 5.1. *This AMR grid shows the mesh refinement at the boundary of a water wave.*

For scientific computing applications, using a hash table simplifies access to the cell information as well as simplifying the performance of various spatial operations. Robey et

al. [16] showed that for every spatial mesh operation, there is an efficient $O(n)$ hash-based algorithm. The simplest method, the perfect spatial hash, avoids the possibility of collisions by directly mapping to a hash table with buckets corresponding to the spatial layout of the grid at the finest level of refinement. At the finest level, each cell is of a size $\triangle_{min}$. For scientific applications on AMR grids with many levels of refinement, it is computationally costly to create and destroy large arrays every timestep. By exploiting the rules of our AMR implementation, we reduce the number of buckets which contain information about the spatial layout of the grid. This reduction yields a sparse, one-dimensional, perfect hash table. This sparse array may be compressed to a compact hash table, but with the cost of dealing with data collisions. This work extends spatial hashing techniques to compact hashing allowing these methods to be viable for large problems on finely resolved meshes. This extension has broad implications in the field of computational science.

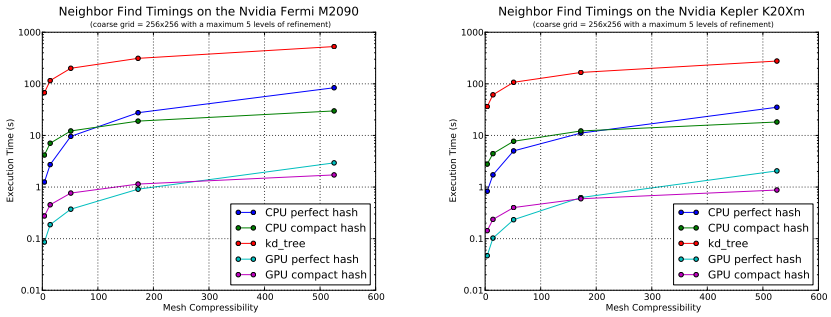
Neighbor determination, a computationally expensive spatial operation in AMR computing methods, can be greatly enhanced using hashing techniques applied to discretized data. As an example, we implement both the compact and perfect spatial hash in a cell-based AMR shallow-water hydrodynamics code (CLAMR). Merging these two methods into a hybrid implementation allows the developer to tailor algorithm performance to the application characteristics and available computing resources. While neighbor determination was the only operation explored in this current work, the properties of computational meshes allow for hash based approaches to be utilized in a variety of scientific applications such as connectivity remapping in Lagrangian hydrodynamics schemes, equation-of-state table look-ups, ray-tracing in Monte Carlo type schemes, just to name a few; the extensions of spatial hashing are ubiquitous in scientific computing. With the ease of parallelizing these methods for multi-core processors and GPU's, spatial hashes provide the computational scientist with a powerful tool for tackling a variety of physical problems.

5.1. Implementation in Hydrodynamics Application. To test our algorithms, we have developed a simple shallow water hydrodynamics scheme capable of running on GPUs and CPUs. The utilization of AMR with GPU portions of the code written in the OpenCL 1.1 standard invited the name CLAMR to be adopted. CLAMR is a second-order accurate hydrodynamics scheme evolved on a cell-based Cartesian adaptive mesh. CLAMR is built on a heterogeneous platform utilizing GPU parallelization in tandem with CPU memory management to achieve significant speed-up and performance. The code was built to be highly versatile allowing us to test and compare the performance of our developed algorithms across a variety of architectures.

We evolve the conservative shallow water wave equations using a total variation diminishing (TVD) finite difference scheme based on a two-step Lax-Wendroff method [10] in conjunction with a minmod symmetric flux limiter developed by Davis [6] and Yee [18] to provide an upwind weighted artificial viscosity term. The Lax-Wendroff method is second-order accurate in space and time, and hence, it is a suitable choice for smooth regions. The wave equations in conservative form are given by,

$$\begin{aligned}
\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} &= 0 \\
\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial}{\partial y}(huv) &= 0 \\
\frac{\partial (hv)}{\partial t} + \frac{\partial}{\partial x}(hvu) + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) &= 0
\end{aligned}$$

where h is the height of a column of water, g is the acceleration due to gravity, and u and v are



a. b.

FIGURE 5.2. *Hashing methods improve neighbor find timings in CLAMR on a) a Fermi GPU and b) a Kepler GPU.*

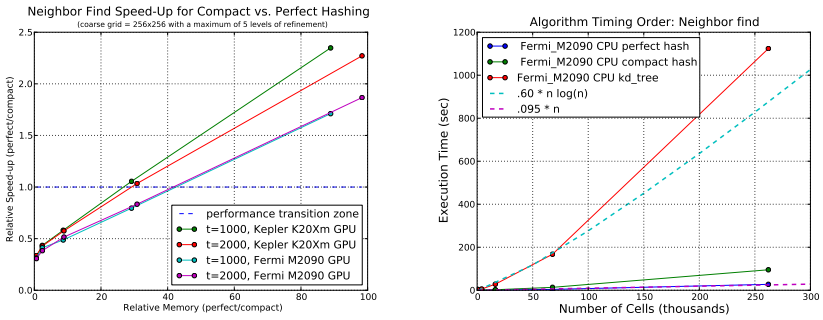
the wave velocities in the x and y directions, respectively. The water is approximated to be incompressible allowing height to be substituted for mass while the pressure is approximated as $gh^2/2$. For a more rigorous presentation, see the sections in Landau and Lifshitz [9] on long gravity waves and shallow-water theory.

We impose several conditions when implementing our cell-based AMR scheme: (i) Two neighboring cells are allowed to differ by no more than one level of refinement so as to reduce errors generated by a reflection of information at cell boundaries during the refinement step. (ii) A cell is refined by symmetrically bisecting along all axes. (iii) Cells are only refined over regions of physical interest where high resolution is necessary to the calculations. (iv) Refinement precedes the arrival of steep gradients or wavefronts. (v) Grid elements are indexed in standard Cartesian coordinates.

The nature of AMR grids will result in patterns of concentrated fine mesh around wavefronts and other areas with steep gradients. Also, refinements will come in blocks of 4 for two-dimensional grids. The randomization of the compact hash is designed to break-up these patterns and reduce collisions in the hash table while saving considerable memory in comparison to the perfect hash.

5.2. Performance Results using CLAMR. Here we explore the performance of perfect and compact spatial hashing across different devices within the context of a hydrodynamics application. We attained timings for determining neighbor cells as a function of the compressibility of the mesh using single node processing on Nvidia's Kepler K20Xm and Fermi M2090 graphics cards and on the 3.5GH intel i7 CPU as shown in Figure 5.2. In these calculations, we began with a coarse grid of 256 x 256 elements and allowed up to five levels of refinement. At five levels of refinement, the perfect hash table can exceed 67 million elements. In the wave simulation only a small circular region around the wavefront is refined and, therefore, the mesh is very compressible. Here we have mesh compressibility calculated as the ratio of the number of cells at the finest level of the mesh to the number of cells in the AMR mesh after $t = 1000$ iterations in the simulation.

The perfect hash achieves higher computational speed at low compressibility, but eventually the compact hash has increased speed and reduced memory usage (see Figures 5.2a, b and Figure 5.3a). This occurs because the compact hash requires extra operations for querying and accessing keys due to data collisions while the perfect hash has higher memory cost for initializing sparse hash tables. As we increase beyond four levels of refinement, the array size of the perfect hash becomes exceedingly large offsetting the costs of queries and data access in the compact hash. In the vicinity of this transition region, the code developer may



a. b.

FIGURE 5.3. *a)Relative memory usage scales linearly with relative performance in compact versus perfect hashing methods. b) The scaling of the hashing method is an improvment upon the scaling of the k-D tree as the problem size increases.*

taylor the method used to suit the specific need of the problem or available computational resources.

The performance of the compact hash is rather remarkable. The compact hash on the CPU is roughly 20x faster than the k-D tree method, and on the GPU it is roughly 200x faster. Another trend to note in Figures 5.2a,b is that the perfect hash has a slope that approaches the k-D tree method. By extrapolating the slope, it becomes evident that the perfect hash will overtake the k-D tree method, becoming slower at some higher mesh compressibility. However, the compact hash has a slope which tracks the k-D tree method. This trend demonstrates the scalability of the compact hash with larger problem sets.

In support of the scalability on the CPUs, we show the $O(n)$ behavior of the perfect hashing method versus the $O(n \log n)$ behavior of the k-D tree method in Figure 5.3b. Here we calculated execution time as a function of number of cells on the coarse grid. Though the compact hash does not exactly scale with the perfect hash, the figure indicates that its behavior is closer to $O(n)$ than to $O(n \log n)$. These calculations were only performed on the Fermi GPUs.

To further demonstrate the memory savings of the compact hashing method, we plot speed-up as a function of memory usage for the perfect hash normalized by the compact hash in Figure 5.3a. Again, we tested on both the Kepler and Fermi GPUs. Due to the symmetry of the spherical wave, a longer iteration corresponds to a larger wave radius, and therefore, the mesh will adapt a larger number of refined cells to capture the steep gradients around the wavefront. We wanted to see how the compact hash performed as the number of refined cells increased on a fixed allowable refinement and so we show the results from simulations at two maximum time iterations, $t = 1000$, and $t = 2000$. We see that the performance improvements of the compact hash scales linearly with the memory usage improvments of the compact hash. For example, when the memory used by the perfect hash is between 20-40x the memory needed for compact hashing, we are crossing the threshold where the compact hash becomes the faster method. Again, the programmer may tailor the method being used to suit their needs.

6. Conclusions. We extended the perfect hash techniques to a highly parallel compact spatial hashing method for determining neighbor cells on an AMR grid and showed that compact hashing offers speed-up and memory savings over the perfect hash. We demonstrated that compact hashing is a memory efficient method of exploiting $O(n)$ algorithms for computational mesh operations. Spatial hashes provide a full range of performance and memory

options for the computational code developer, allowing scientific applications to exploit the performance enhancements of hashing while operating over diverse ranges of resolution in large problem sets.

We applied spatial hashing methods on AMR grids with 67 million elements at the finest level of refinement and our results showed a 20x speed-up on the CPU compared to a k-D tree method and 200x speed-up over the CPU k-D tree on every GPU device tested. Beyond 4 levels of refinement, the compact hash is faster than the perfect hash and reduces the memory usage by approximately a factor of 30. Hybridization of the perfect and compact methods gives the developer the ability to tailor the method used to the constraints on time or available resources. The thread-based parallelism of the compact hash method allows performance portability on CPU and GPU architectures. CLAMR demonstrates the use and performance of hashes for applications on future architectures. The application of spatial hashing methods are ubiquitous in scientific computing.

Future avenues. Hash tables can be employed for spatial operations such as sorts, remaps, and table lookups; neighbor determination is only one operation which benefits from the methods we have developed. In the future, data sparsity could be exploited in these operations to allow the use of a compact hash. These improved algorithms could provide a broader model for developers to follow when tailoring methods to suit their specific application. The scalability of hash algorithms in parallel computing with more distributed memory could be evaluated. Further work on the hash library could also be pursued. Although we implemented many standard hashing optimizations in the library, there are plenty of other optimizations that could be explored. The performance of these optimizations and their compatibility with the hash-based algorithms in the massively parallel environment could be studied.

Acknowledgements. We would like to acknowledge our families who have been continuous sources of encouragement. A special thanks to Scott Runnels for organizing the wonderful Computational Summer Physics Workshop, and Los Alamos National Lab for hosting us. We owe homage to the technical wizards who protect and maintain Darwin, our favorite experimental architecture computer in CCS-7. Much thanks to Rachel Robey for offering her wisdom and experience in the matters of hashing. Thanks to the LANL Director's Office for partial funding of this work.

REFERENCES

[1] D. A. ALCANTARA, *Efficient Hash Tables on the GPU*, PhD thesis, UC Davis, 2011.
[2] D. A. ALCANTARA, A. SHARF, F. ABBASINEJAD, S. SENGUPTA, M. MITZENMACHER, J. D. OWENS, AND N. AMENTA, *Real-time parallel hashing on the GPU*, ACM Trans. Graph., 28 (2009), pp. 154:1–154:9.
[3] J. R. BELL, *The quadratic quotient method: a hash code eliminating secondary clustering*, Communications of the ACM, 13 (1970), pp. 107–109.
[4] J. L. CARTER AND M. N. WEGMAN, *Universal classes of hash functions*, Journal of Computer and System Sciences, 18 (1979), pp. 143–154.
[5] Z. J. CZECH, G. HAVAS, AND B. S. MAJEWSKI, *Perfect hashing*, Theoretical Computer Science, 182 (1997), pp. 1–143.
[6] S. F. DAVIS, *A simplified TVD finite difference scheme via artificial viscosity*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. 1–18.
[7] H. GAO, J. F. GROOTE, AND W. H. HESSELINK, *Efficient almost wait-free parallel accessible dynamic hashtables*, tech. rep., CS-Report 03-03, Eindhoven University of Technology, The Netherlands., 2003.
[8] D. E. KNUTH, *Sorting and Searching, vol. 3*, Addison-Wesley, 1973.
[9] L. LANDAU AND E. LIFSHITZ, *Fluid mechanics. 1987*, Course of Theoretical Physics, (1987).
[10] P. LAX AND B. WENDROFF, *Systems of conservation laws*, Communications on Pure and Applied Mathematics, 13 (1960), pp. 217–237.

[11] W. D. MAURER, *Programming technique: An improved hash code for scatter storage*, Communications of the ACM, 11 (1968), pp. 35–38.

[12] M. MITZENMACHER AND S. VADHAN, *Why simple hash functions work: exploiting the entropy in a data stream*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2008, pp. 746–755.

[13] J. K. MULLIN, *A caution on universal classes of hash functions*, Information Processing Letters, 37 (1991), pp. 247–256.

[14] J. I. MUNRO AND P. CELIS, *Techniques for collision resolution in hash tables with open addressing*, in Proceedings of 1986 ACM Fall Joint Computer Conference, IEEE Computer Society Press, 1986, pp. 601–610.

[15] W. W. PETERSON, *Addressing for random-access storage*, IBM Journal of Research and Development, Vol. 1 (1957), pp. 130–146.

[16] R. N. ROBEY, D. NICHOLAEFF, AND R. W. ROBEY, *Hash-based algorithms for discretized data*, SIAM Journal on Scientific Computing, 35 (2013), pp. C346–C368.

[17] F. A. WILLIAMS, *Handling identifiers as internal symbols in language processors*, Communications of the ACM, 2 (1959), pp. 21–24.

[18] H. YEE, *Construction of explicit and implicit symmetric TVD schemes and their applications*, Journal of Computational Physics, 68 (1987), pp. 151 – 179.

for a single $g_{n,\text{ic}}$ coefficient across a work-group. If there are not enough terms for all of the work-items, we lose much of the benefit of parallelization on the GPU, since some of the threads then do no work. One way to remedy this in practice is to use smaller work-group sizes when dealing with smaller polynomials. However, this solution has limitations since GPUs are most efficient when the work-group size is a multiple of 32 [6]. A similar drawback arises if sparsity is not exploited, where the GPU sees a ∼2x slow-down for 3$^{\text{rd}}$ order polynomials. This results from parallelizing along the tensor multiplication loop, where many threads end up multiplying zeros together and appending them to a sum, again wasting effort.

Second, the data indicates the GPU is increasingly advantageous as it is given more work. As the degrees of freedom and number of operations increases, whether from refining the grid or increasing the number of polynomials, the speedup increases. This reflects the streaming memory model on the GPU, where floating-point operations are almost free.

Using OpenCL event timers, we can further break down the GPU execution time, to investigate where the GPU is spending most of the execution time. This is shown in Table 3 for the degree 3 polynomial, 40x40x40 grid case. As

Table 3: GPU Event Timing

Event	Time (ms)
Kernel Create	0.1628
Data Send	336.9
Compute	4763
Data Receive	20.14
Total	4784

a perhaps unexpected result, the computations overwhelmingly dominate the execution time. In other applications, memory transfer operations take up a significant portion of the runtime. However, this case involves a high work to data ratio, since the method involves a high number of operations relative to the amount of data transferred. As a result, optimizations that focus on decreasing compute time are more beneficial than memory optimizations, contrary to the usual case for GPU algorithms.

Note: these times may overlap, so the total compute time is not necessarily the sum of event times.

6 Concluding Remarks

This work has demonstrated that taking advantage of sparsity, whenever possible, is crucial to developing efficient algorithms, especially on the GPU. Furthermore, an overall speedup ranging from 2-60x for GPUs over CPUs was found, advocating the applicability and benefit of GPUs in numerical algorithms, especially for independent segments of code that benefit from parallelism. These speedups indicated that more work given to the GPU results in more speedup, especially when increasing the number of basis functions used. This compliments the results in [4], where it was found that the discontinuous Galerkin conservative level set method is more effective when more basis functions are used. Therefore, accelerating that method via GPUs reaps benefits from multiple angles, making it an excellent candidate for high order interface tracking and modeling atomization.

Future work to further develop this approach would involve an improved implementation, for instance, ensuring memory alignment. Porting more segments of code to OpenCL would also be valuable. This is especially true when there is a direct benefit from GPU architectures, but is also true for code that would not be improved by parallelism, since it avoids passing data to and from the GPU as much as possible. For instance, it may be beneficial to handle ghost cell updates on the GPU, allowing multiple time steps to be executed before returning to the CPU. Finally, to complete the conservative level set method it is necessary to implement a similar scheme for reinitialization, Eq. (6).

$$\frac{\partial G}{\partial \tau} + \nabla \cdot (G\,(1 - G)\,\hat{\boldsymbol{n}}) = \nabla \cdot (\varepsilon\,(\nabla G \cdot \hat{\boldsymbol{n}})\,\hat{\boldsymbol{n}}) \tag{6}$$

Reinitialization is best described as a nonlinear companion to the advection equation solved in pseudo-time which greatly improves mass conservation.

Acknowledgments

I would like to acknowledge the support of my mentor, Bob Robey, for his invaluable advice and teaching through the summer. I would also like to thank Scott Runnels, who organized the Los Alamos National Laboratory Computational Physics Student Summer Workshop 2013, where this work was performed. Finally, I would like to thank the teams managing the Darwin CCS-7 experimental cluster and the Moonlight ASC cluster, where these algorithms were tested and benchmarked.

References

[1] B. Cockburn and C.-W. Shu. "Runge–Kutta discontinuous Galerkin methods for convection-dominated problems". *J. Sci. Comput.* 16 (2001), pp. 173–261.

[2] I. Duff, R. Grimes, and J. Lewis. *User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)*. 1992.

[3] M. Herrmann. "A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids". *J. Comput. Phys.* 227 (2008), pp. 2674–2706.

[4] Z. Jibben and M. Herrmann. "An arbitrary high-order conservative level set Runge-Kutta discontinuous Galerkin method for capturing interfaces". *ILASS Americas 25th Annual Conference on Liquid Atomization and Spray Systems* (2013).

[5] P. LeSaint and P. A. Raviart. "On a finite element method for solving the neutron transport equation". In: *Mathematical Aspects of Finite Elements in Partial Differential Equations*. Ed. by C. de Boor. Academic Press, NY, 1974, pp. 89–123.

[6] *NVIDIA OpenCL Best Practices Guide*. ver. 1.0. NVIDIA Corporation. 2009.

[7] E. Olsson and G. Kreiss. "A conservative level set method for two phase flow". *J. Comput. Phys.* 210 (2005), pp. 225–246.

[8] E. Olsson, G. Kreiss, and S. Zahedi. "A conservative level set method for two phase flow II". *J. Comput. Phys.* 225 (2007), pp. 785–807.

[9] *The OpenCL Specification*. ver. 1.2. Khronos Group, Inc. 2011.

Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo

(Forrest Brown, mentor)

LA-UR-13-26449
Approved for public release; distribution is unlimited.

Title: Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo

Author(s): Josey, Colin
Veit, Max D.

Intended for: MCNP documentation
Report
Web

Issued: 2013-08-15

Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo

Colin Josey, Max Veit

Computational Physics Workshop
Los Alamos National Laboratory

August 14, 2013

Abstract

Monte Carlo criticality calculations for nuclear reactors have typically only been able to find the primary eigenvalue and eigenmode. With the new fission matrix capability in MCNP6, it becomes possible to solve for higher modes, which are useful in wide variety of analyses. As a test case of this new capability, two reactor models in both a base configuration and a perturbed configuration were simulated in MCNP to generate fission matrices. Tools were written to find a relatively small number of eigenpairs of the resulting matrices, whereupon it was found that the implicitly restarted Arnoldi method outperformed the previously used power iteration as an eigenvalue algorithm by a factor of 1500 for a 3600×3600 sparse matrix. The primary eigenvalue was then compared to the default MCNP result and, although the values showed bias with regards to mesh size, the matrix-derived values had superior statistics. With the eigenvalue verification complete, the primary eigenmode of the base case was then projected onto the perturbed core's eigenspace, where transition coefficients, simplified quasistatic transitions, and projection errors were calculated. The projection error typically dropped off after the first 20 eigenvalues to a value that was stable through the first 100.

1 Introduction

Criticality calculations using the MCNP Monte Carlo code determine the fundamental mode eigenvalue (k_{eff}) and eigenfunction (fission distribution) of a fissile system. These calculations are routinely used in the design and analysis of critical experiments, nuclear reactor cores, and criticality safety applications. Recently developed MCNP capabilities for the fission matrix method permit the calculation of higher-mode eigenvalues and eigenfunctions. With knowledge of the higher modes, transitions from the base state of a system to perturbed states can be analyzed. The state transition parameters characterize changes to the system induced by material substitutions, geometry changes, and feedback, and are important for analyzing potential instabilities.

The fission matrix is essentially a spatially discretized Green's function for the neutron transport equation. It is a matrix $\bar{F}$ whose entries, $\bar{F}_{ij}$, contain the expected number of next generation fission neutrons generated in spatial mesh region i by a neutron born in mesh region j. The eigenvalues and eigenvectors of this matrix will approximate the eigenvalues and eigenvectors of the reactor. These approximations are mesh size dependent, but the size of the matrix goes as the square of the number of cells. For even moderate mesh sizes, the resulting fission matrix will be massive. Thankfully, it is also rather sparse. In this report the methods of solving for the eigenvalues and eigenvectors of a very large sparse matrix are detailed. Tools that analyze the output of the fission matrix from MCNP were implemented in C++, MATLAB and Python.

Demonstrations of the capabilities of the fission matrix were performed on two reactor models, the Advanced Test Reactor, and a 2D PWR model. These models are frequently used in the testing of MCNP. These models were then perturbed and the resulting eigenpairs were analyzed.

Lastly, as it is very difficult to store all of the data for the fission matrix itself, additionally storing the squares of the tallies that generated it to calculate statistics is not implemented as of yet. By using a moderate quantity of MCNP runs with varying starting seeds, the statistics can roughly be approximated. The same two models used for testing MCNP as before were used here as well.

2 Theory

First, the theory and physics underlying the concept of fission matrices is summarized, and the linear algebra behind transition coefficients investigated. The predominant algorithms used for finding eigenpairs of large, sparse, asymmetric matrices are also listed and briefly discussed.

2.1 Fission Matrices

The utility of a fission matrix is rooted in the neutron transport equation. Through no approximation other than a simple spatial discretization, the neutron transport equation

$$M\Psi(\mathbf{r}, E, \hat{\Omega}) = \frac{1}{k}\frac{\chi(E)}{4\pi}S(\mathbf{r}) \tag{2.1}$$

can be integrated over space and energy into the form

$$\mathbf{s} = \frac{1}{k}\bar{F}\mathbf{s}, \tag{2.2}$$

where $\mathbf{s}$ is the source distribution vector and $\bar{F}$ is the fission matrix as defined before [1]. As shown, it is clear that this is an eigenvalue problem, with k as the eigenvalue and $\mathbf{s}$ as the eigenfunction. As all components are matrices, vectors, or scalars, this equation lends itself well to a linear algebra solution.

The solutions of this eigenvalue problem are especially useful for a specific type of analysis, namely, computing transition coefficients for reactor perturbations. First, assume the reactor configuration instantaneously changes, through e.g. a geometry or a material property change. In this situation, one would like to know how the reactor's initial fission source distribution can be represented in terms of eigenmodes of the fission matrix of the new configuration. In particular, one would like to be able to express any of the initial configuration's eigenmodes in terms of this new partial basis.

Let $\{\mathbf{u}_i\}$ be the set of eigenmodes of the fission matrix of the initial configuration, and $\{\mathbf{v}_i\}$ be the set of eigenmodes in the final configuration. Assuming the fission matrix for the final state has at least m linearly independent eigenmodes $\{\mathbf{v}_i\}_{i=1}^{m}$, then any source distribution $\mathbf{s}$ can be approximated by its projection onto the space spanned by these m modes. This approximation can be considered optimal (in

the sense that it minimizes the norm of the residual) if the set of right eigenmodes is orthogonal.

Applying the above expansion to the i-th eigenmode of the initial fission matrix, $\mathbf{u}_i$, gives:

$$\mathbf{u}_i \approx \mathbf{u}_i^m = \sum_{j=1}^{m-1} C_{ij}\mathbf{v}_j. \tag{2.3}$$

The j-th expansion coefficient C_{ij} can be extracted by exploiting the fact that the system of forward and adjoint modes $\left\{\left(\mathbf{s}_k, \mathbf{s}_k^\dagger\right)\right\}_{k=0}^d$ of a diagonalizable matrix forms a biorthogonal system; i.e., with the appropriate normalization,

$$\left\langle \mathbf{s}_l, \mathbf{s}_m^\dagger \right\rangle = \delta_{lm}. \tag{2.4}$$

The notation $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product on $\mathbb{C}^n$, which for the purposes of this report is defined as $\mathbf{b}^*\mathbf{a}$. This product is linear in the first argument and conjugate-linear in the second, a property called 'sesquilinearity'.

The adjoint modes are the left eigenvectors of the fission matrix. In practice, they can be computed by taking the eigenvectors of its transpose. In the general case, the resulting vectors are the complex conjugates of the adjoint modes. This is usually not a problem, since it is usually assumed that the eigenmodes of a fission matrix will be real. Although this fact has not been theoretically proven, numerical evidence bears it out within the limits of statstical variation [1].

Relation (2.4) holds, albeit in a restricted sense, even if a complete system of eigenvectors does not exist for a given fission matrix; see Appendix A. Using this fact, one can write, using the linearity of the dot product in its first argument,

$$C_{ij} = \left\langle \mathbf{u}_i, \mathbf{v}_j^\dagger \right\rangle = \sum_{k=1}^{N} C_{ik} \left\langle \mathbf{v}_k, \mathbf{v}_j^\dagger \right\rangle = \sum_{k=1}^{N} C_{ik}\delta_{kj}, \tag{2.5}$$

assuming the final eigenmodes are normalized such that Relation (2.4) holds.

The approximation $\mathbf{u}_i^m$ can then be constructed using the transition coefficients $\left\{C_{ij}\right\}_{j=1}^m$ and the eigenvectors $\left\{\mathbf{v}_j\right\}_{j=1}^m$.

Once these transition coefficients are known, a quasistatic model of the transition from the base to the perturbed state is:

$$\mathbf{s}_{\text{current}} = \sum_{j=1}^{N} C_{1,j} \left(\frac{k_j}{k_1} \right)^n \mathbf{v}_j, \tag{2.6}$$

where n is the current neutron generation and N is the total number of eigenvalues.

2.2 Eigenvalue Solvers

The fission matrix $\bar{F}$ is nonsymmetric and tends to be a large, sparse matrix. As such, it is very difficult to store without using a sparse storage scheme. This severely constrains the choice of eligible algorithms. For example, the implicit QR algorithm, the eigenpair solver of choice for dense matrices, requires a transformation into upper Hessenberg form [2]. Such transformations tend to cause fill-in of the formerly empty elements, causing what was once a sparse matrix to occupy more than half of the space of a full version of the matrix. Further, the resulting matrix of eigenvectors will be nearly full. Most computers cannot handle matrices so large.

An alternative comes in the form of Krylov subspace solvers. The primary advantage of such solvers is the iterative mode of calculation that performs only matrix-vector math with the sparse fission matrix, so the problem of fill-in is avoided. The most simple Krylov subspace solver is the power iteration method. This simple algorithm is robust and effective, but has two major flaws. For one, to get more than the first eigenpair requires some sort of deflation, such as Hotelling deflation [3, pp. 85-96]. Secondly, the convergence of the power method is linear and governed by the factor $|\lambda_1/\lambda_2|$, also known as the dominance ratio [4]. When this ratio is close to one, the convergence of this method can be extremely slow.

More advanced Krylov subspace solvers such as implicitly restarted Arnoldi method (IRAM) address both issues. IRAM can solve for more than one eigenpair at a time and it has (usually) superlinear convergence [5]. All math done with regards to the fission matrix is in matrix-vector form. Two matrices are stored separately from the fission matrix. Matrix V has columns of the same length as the fission matrix, but a number of rows equal to m, a variable that is between 2-3x as large as the total number of eigenpairs needed. The columns

are, once sorted, the resulting eigenvectors of the problem. Matrix H is an upper Hessenberg matrix of size m by m whose eigenvalues are the eigenvalues of the fission matrix [3, pp. 128-136].

3 Methodology

Several steps were involved in the investigation of the fission matrix capability. First, in order to study transitions and transition coefficients, reactor models were needed in both an initial and final state. Then, a tool was needed to find the eigenpairs of both systems efficiently. These tools were finally expanded to perform numerous data manipulations and plot the results.

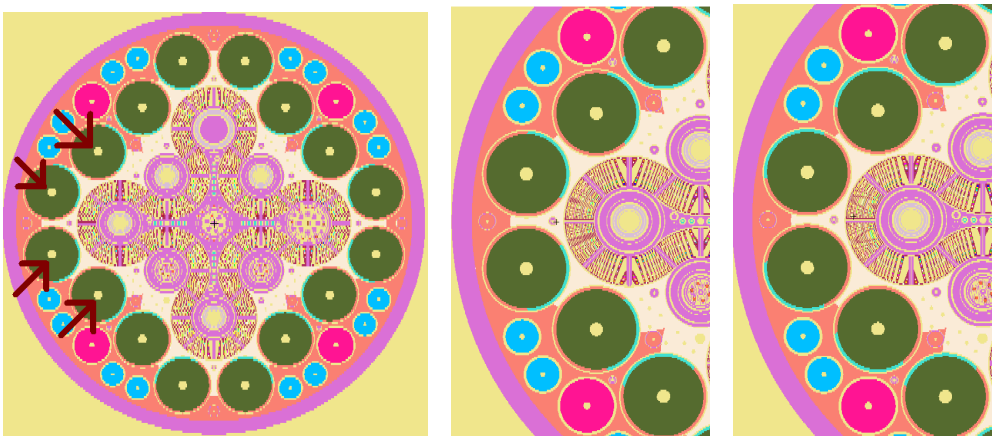
3.1 Reactor Model Modification

A main goal of the project was to test the fission matrix capability on a reasonably complex analysis problem. The problem of finding transition coefficients between two reactor models was chosen for this purpose. This analysis requires perturbed versions of base-case reactor models, as well as the base models themselves. If the two models were too different, the transition coefficients would be essentially meaningless, so the alterations were done in such a way to significantly alter the flux while at the same time not significantly altering the geometry. For the ATR case, four of the control drums S3, S4, W1, and W2 were rotated 50°, moving beryllium closer and hafnium away from the core. The core model, before and after, is shown in Figure 1.

For the 2-D PWR core, control rods consisting of type 304 stainless steel were inserted in each assembly in the upper right quadrant of the core. A comparison is also shown in Figure 2. Note that the region chosen is asymmetric in that it does not represent a true rotationally symmetric quarter of the core; this fact has significant consequences for the eigenmodes of the perturbed case as well as the transition coefficients in between the states.

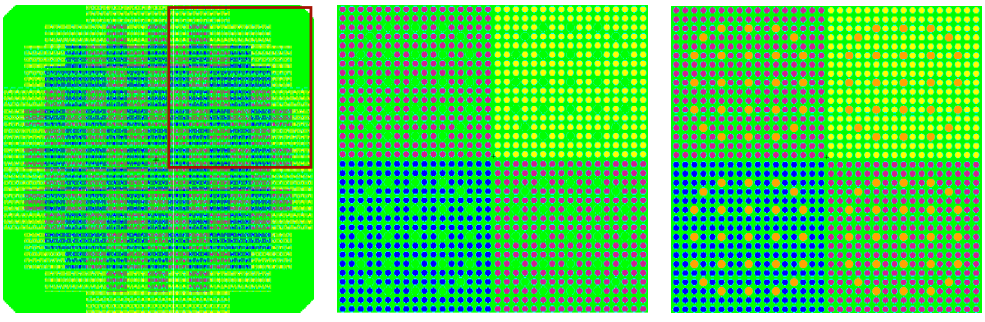
3.2 Fission Matrix Generation

Currently, as of August 2013, published versions of MCNP6 do not contain the full fission matrix capability. The commands described



(a) ATR Full Core (b) ATR Original (c) ATR Modified

Figure 1: ATR Core Modifications



(a) PWR Full Core (b) PWR Original (c) PWR Modified

Figure 2: PWR Core Modifications

below will work, but no fission matrix will be output. Further, as time goes on, the method of generating fission matrices as well as their output format will likely change as they are not precisely user friendly at the moment. This part of the report will detail how current versions of MCNP6 internal to LANL operate with regards to fission matrices.

In order to enable fission matrices, two cards must be present in the input deck. First, the mesh extent from the `HSRC` card is used as the extent for the fission matrix. As such, the `HSRC` card must be included. The number of cells along each axis will not be used, however, as the fission matrix usually needs to be finer than the Shannon entropy mesh. The second is the `KOPTS` card. The `KOPTS` card defines the options for `KCODE`, and contains the settings for the fission matrix. `KOPTS` has the following form:

```
KOPTS   ...several-unrelated-options...
        fmat= (yes/no)
        fmataccel= (yes/no)
        fmatskip= n
        fmatncyc= n
        fmatnbr=  n
        fmatnbrx= n
        fmatnbry= n
        fmatnbrz= n
```

The meaning of these options is summarized in Table 1.

As long as both commands are present, MCNP will keep the fission matrix tallies internally. If `fmataccel = yes`, the resulting primary eigenfunction from solving the fission matrix will be used to split or roulette `KCODE` source neutrons in each cell to closer match the source distribution. This has some advantages, as the fission matrix primary eigenmode will be more accurate than the initial guess or any unaltered `KCODE` results prior to convergence. Current versions of MCNP do not save the fission matrix, but in-development versions do. The resulting file is `fmat_file`. The structure of this file as currrently implemented is discussed in the following sections.

3.3 Eigenvalue Tool Exploration

By default, the current development versions of MCNP6 output the fission matrix as an unformatted binary file as written by a Fortran-

command	Description
fmat= yes	Enables fission matrix
fmataccel= yes	Enables using the fission matrix primary eigenmode to accelerate the convergence of KCODE
fmatskip= n	Skips n cycles before tallying fission matrix
fmatncyc= n	Solves for k_{eff} and dominance ratio every n cycles
fmatnbr= n	Total number of entries available for the sparse matrix.
fmatnbrx= n	Sets number of cells in the x-axis to n
fmatnbry= n	Sets number of cells in the y-axis to n
fmatnbrz= n	Sets number of cells in the z-axis to n

Table 1: KOPTS Options

based code. These files are not human-readable, and their structure is machine and compiler dependent. A Fortran-based tool already existed that would read the fission matrix, perform power iteration, and output several text and PostScript files containing results and plots. This worked well on smaller fission matrices on the order of $N = 10000$ rows, but beyond that, the calculation time was unreasonably high. Two different approaches (with a third later) were undertaken to expedite the calculation process. The first approach used Python interfaced with ARPACK [6], the second was a custom C++ implementation of Arnoldi iteration, and the third used MATLAB, again interfaced with ARPACK.

Each toolset had in common the same general steps necessary to perform the analysis: First, it read in the initial and final (perturbed) fission matrices from their native binary formats. Second, the tool normalized each matrix's rows against the corresponding fission source tally, since the matrices were saved in the form of raw tallies. Third, it extracted the forward and adjoint eigenvalues of the resulting sparse matrix. Finally, it obtained transition coefficients by taking dot products between the initial forward and final adjoint eigenmodes. These results were then visualized and interpreted.

3.3.1 Python

One approach to extracting the eigenmodes from the fission matrices used an interactive Python analysis system[1] equipped with several useful libraries. The system used the NumPy package for numerical computing, as well as the SciPy package's sparse-matrix capabilities, which included an interface to the freely available ARPACK sparse-matrix eigenvalue solver [6]. Plots and visualizations were generated using the Matplotlib visualization package.

The system proved effective at reading in the generated fission matrices in Fortran unformatted-binary sparse storage format and finding a small (80 or fewer) number of eigenvectors and -values. As an illustrative example, the eigensolver routine took 163 seconds to solve for the 80 forward modes of the unperturbed case with $N = 57600$; the adjoint case usually took longer (212 seconds in this case)

[1]Technical details: Enthought Python Distribution 7.3-2 (64-bit) (`https://www.enthought.com/`) with Python 2.7.3, SciPy 0.10.1, NumPy 1.6.1, Matplotlib 1.1.0, and IPython 0.12.1 for interactive use.

because of the additional cost of multiplying by the transposed fission matrix; see Section 4.1.

The prime disadvantage of the Python system is that it did not have parallel processing capability in the standard configuration. A parallel implementation of ARPACK is available [6]; however, the SciPy package apparently only links to the serial version. Moreover, the sparse matrix-vector products themselves must be performed by SciPy; this procedure is apparently also implemented serially in standard configuration.

3.3.2 C++

Initially, the C++ code, called `eigtest`, was simply a matrix math library that would implement just enough matrix math to do power iteration so that the person writing it could relearn the programming language. After finding out about the massive speedup possible converting from power iteration to the implicitly restarted Arnoldi method, the tool was converted to mimic a reference implementation written in MATLAB [7]. Portions of the original F90 code were merged in to provide for reading the fission matrix files. After a few weeks, however, the implementation was not converging correctly, and due to limited time, it was scrapped. The ability for the tool to read and write the matrix in a few different formats proved useful later on in the MATLAB implementation.

3.3.3 MATLAB

Although it may seem a bit strange to implement this code in both Python and MATLAB, during the C++ implementation, a large amount of MATLAB backend was written before the Python implementation was complete. In general, the MATLAB capability is very similar to the Python one and was used for the core step-by-step transition calculations and the statistics[2].

3.3.4 Fortran ARPACK Interface

Near the end of the project, another interface was developed for ARPACK, written in Fortran 90 and using parallel sparse-matrix vector

[2]Technical details: Matlab version R2013a

products. By that time, however, most of the work requiring eigenvalue solvers was completed, so it was not extensively tested.

3.4 Verification

It is not clear to what extent the algorithms and implementation of ARPACK have been verified. Therefore, two simple independent checks were undertaken to verify that ARPACK was indeed returning results with the advertised properties.

The nature of the eigenvalue problem admits a straightforward method of verification: Given a computed solution $(\mathbf{v}_C, \lambda_C)$ to the problem $A\mathbf{v} = \lambda\mathbf{v}$, the residual $r = A\mathbf{v}_C - \lambda_C\mathbf{v}_C$ can be computed, and its properties (e.g. norm) investigated to assess whether the computed answer solves the problem to the desired tolerance. A residual was considered acceptable if its l_2 norm did not considerably exceed the number of mesh cells used times the machine epsilon. This was indeed found to be the case for a set of 80 eigenvalues extracted both from the PWR and the ATR cases. We suspect this check is already done internally in ARPACK, but since no concrete evidence of such a check could be found, an independent verification was seen as justified. This independent check also guarded against any possible failure conditions in ARPACK that would not have been reported back to the user.

Another important criterion on the set of eigenvectors returned by ARPACK is that they be linearly independent; this property can be checked using a singular value decomposition of the matrix of eigenvectors. For both the PWR and the ATR test cases, it was indeed the case that all 80 singular values of the matrix of eigenvectors were clearly nonzero, i.e. many orders of magnitude larger than machine precision.

4 Results

The two different test problems, PWR and ATR, were run in MCNP with different parameters. A summary of the parameters used is in Table 2. These two runs generated fission matrices that were 4.8 GB and 32 MB in size for the PWR and ATR respectively. Both runs had enough cycles discarded to be converged for `KCODE`. The choice of fission matrix mesh size depends on the reactor being studied as well

Reactor	2D PWR	ATR
Cycles	300	500
`KCODE` Active Cycles	200	400
`fmatskip`	3	3
`fmatnbrx`	480	100
`fmatnbry`	480	100
`fmatnbrz`	1	1
Neutrons/cycle	4 million	100 thousand

Table 2: MCNP6 run details

Matrix Size	IRAM (ARPACK)	Power Iteration
3600×3600	3.09 s	4878 s
900×900	0.234 s	353 s
225×225	0.0337 s	30.6 s

Table 3: IRAM vs. Power Iteration for Various Matrix Sizes

as computational resources. For example, the PWR has low connectivity between distal regions of the core, owing to its great size. This is reflected in the fission matrix as a very high sparsity and thus, low memory usage. The opposite case is true for the ATR. As such, the ATR was typically run with a smaller mesh. A smaller mesh also requires fewer neutrons for statistical reasons. However, it is always beneficial to calculate with as large a mesh as possible, because the fission matrix can be aggregated into a smaller one as needed.

In practice, the fission matrix used for the PWR was aggregated by a factor of two, reducing it in size from 230400 to 57600 rows. This made the eigenvalue solve times much more tractable and improved the statistical properties of the resulting eigenmodes.

4.1 Solver Timings

A quick speed comparison was done between the two algoritms available. The runs were done on 1 CPU, and the first 16 eigenmodes and 16 adjoint eigenmodes were solved for. These runs are summarized in Table 3. As shown, a speedup of approximately 1500-fold was ex-

Number of	Unperturbed		Perturbed	
Modes	Forward	Adjoint	Forward	Adjoint
20	95 s	110 s	120 s	149 s
40	129 s	170 s	105 s	143 s
80	163 s	212 s	148 s	196 s

Table 4: Timings of the IRAM eigenvalue algorithm, as implemented by SciPy (see Section 3.3.1), on the PWR problem with $N = 57600$ rows.

perienced in switching algorithms on the 3600×3600 matrix.

Additionally, a preliminary investigation was done to investigate how the time taken by Arnoldi iteration scales with mesh size and the number of modes requested. The results are summarized in Table 4, from the PWR problem with $N = 57600$ rows. The times were computed both for the forward and adjoint mode solves. The given times are best out of 3 runs on a fairly capable multi-user machine. An important caveat on these timings is that a surprisingly large amount of variation was observed in the eigenvalue-solve timings, particularly on a spatial mesh with $N = 230400$.

Since not enough time was available to study the statistical properties of these timings in more detail, they are presented here only in order to illustrate some interesting general properties of the algorithm's runtime. First, note that the algorithm took consistently more time solving the adjoint problem than the forward problem; this is likely due to the overhead involved in transposing a CSR-stored sparse matrix, or computing matrix-vector products with its transpose. Second, the scaling as a function of number of eigenmodes requested is much weaker than one would expect given the known complexities of the individual components of the algorithm. This fact hints at the nontrivial convergence properties of the implicitly restarted Arnoldi method, as does a surprising result where it took longer to solve for 40 modes than for 80 modes of the matrix of the PWR problem with $N = 230400$ rows (1130 seconds versus 793 seconds, best of 6 runs each).

4.2 Eigenmodes

Using the techniques described earlier, the eigenpairs of the resulting matrices were solved for and plotted. Figure 3 shows the first 4 eigenmodes and adjoint eigenmodes of the initial and final configuration of the core. The same was done for the ATR in Figure 4. Comparing the original to the modified for the PWR, it becomes clear that the insertion of the control rods has significantly depressed a quadrant of the fundamental mode. The slight asymmetry in the perturbation also becomes noticeable in the higher modes. As for the ATR, the lobe that is surrounded by the moved control barrels is more active than the rest of the core. This is essentially as expected.

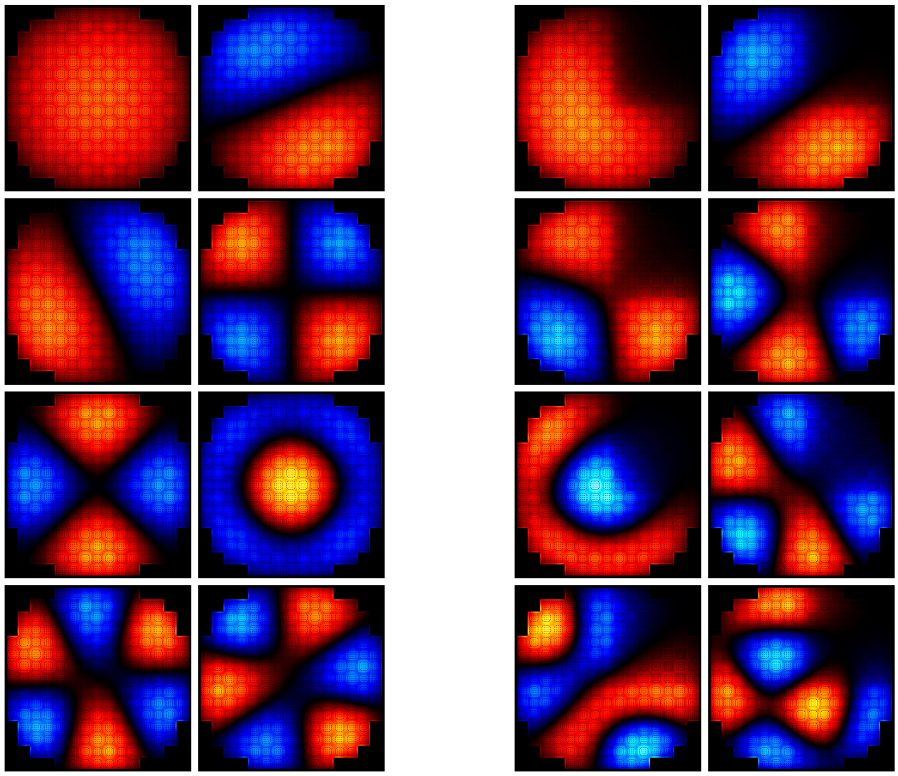
4.3 Transition Coefficients

Using a set of 40 eigenmodes calculated from each fission matrix, the transition coefficients were calculated and mapped into a grid. The PWR and ATR transition coefficients are plotted in Figure 5. The PWR transition coefficients give hints as to the more intricate transition occuring. The insertion of control rods in an asymmetric region of the core is not nearly as uniform an alteration as moving four adjacent control barrels by the same amount, and has a very strong spatial dependence.

4.4 Transitions

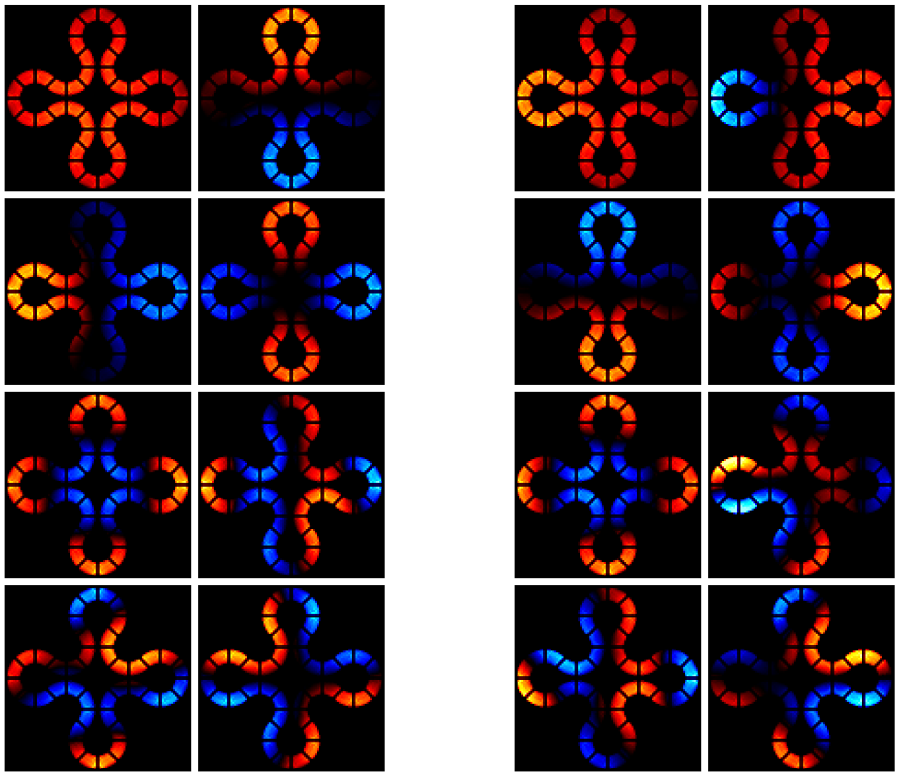
For the ATR, these coefficients were used to reconstruct a very basic stepwise transition. This is shown in Figure 6, along with what should be the original and final source distributions. Very slight differences are evident between the original and the generation 0 result, hinting that the reconstruction is not exact due to the limited number of eigenmodes used. Had the entire set of eigenvalues been calculated, this reconstruction would not be as imprecise.

The transition is found to be relatively smooth with the majority of sources propagating into the lobe of the core with the rotated control drums. The transition is mostly completed after 15 neutron generations, which, ignoring delayed neutrons and other reactions such as temperature dependence that will alter the transition, is a very short time. These plots can be seen as an approximation of the prompt jump at the beginning of a reactor configuration change.



(a) PWR Original (b) PWR Modified

Figure 3: Eigenvectors of the 2D PWR



(a) ATR Original (b) ATR Modified

Figure 4: Eigenvectors of the ATR

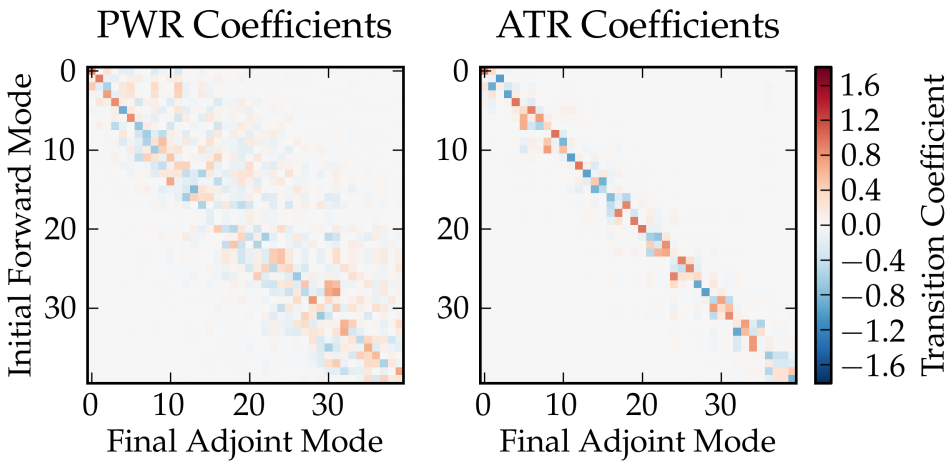


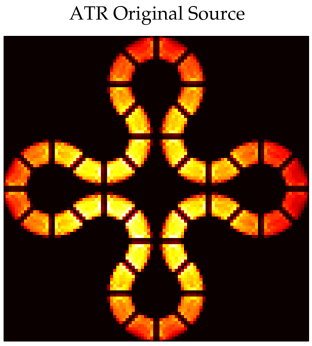
Figure 5: Transition Coefficients

4.5 Reconstruction Error

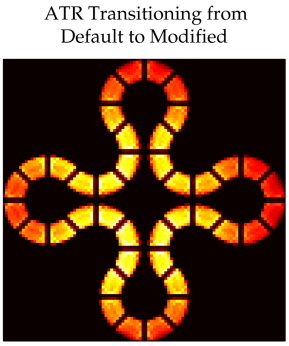
The error between the fundamental mode and its reconstruction in the space spanned by the perturbed eigenmodes was also studied. The initial fundamental mode was reconstructed with various quantities of eigenmodes in the final configuration and the l_2 norm of the difference between the two was calculated. The differences for both reactors with 40 eigenmodes are plotted in Figure 7, with the eigenmode count dependence plotted in Figure 8. It is clear that the more symmetric perturbation in the ATR core has made the transition coefficients rather simple. It takes relatively few transition coefficients to properly reconstruct the initial modes. The rather complex perturbation done to the PWR causes a more spread-out set of transition coefficients, indicating the larger number of perturbed modes necessary to reconstruct any given initial mode.

4.6 Statistics

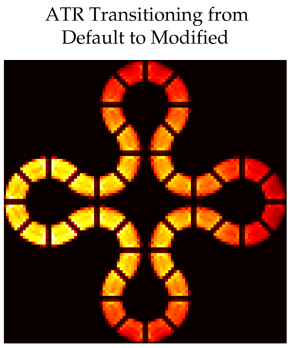
One primary drawback to the fission matrix method is that it is already quite difficult to store even rather small fission matrices. As



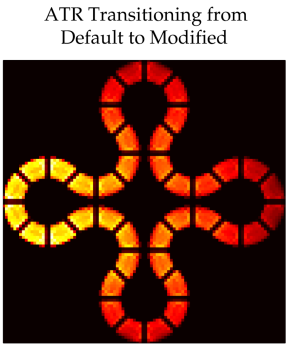
(a) ATR Original



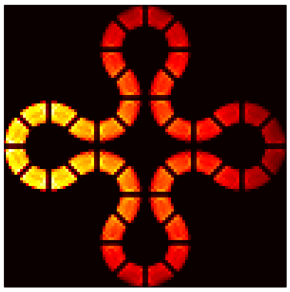
(b) ATR Generation 0



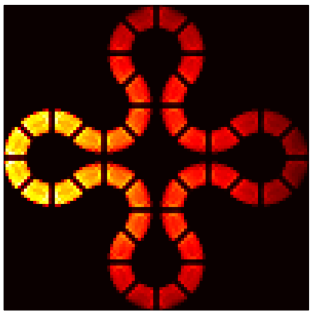
(c) ATR Generation 5



(d) ATR Generation 10

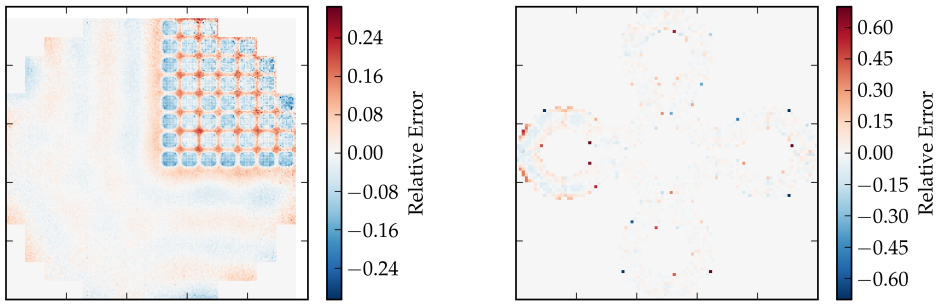


(e) ATR Generation 15



(f) ATR Final

Figure 6: ATR Transition from Start to Final



(a) PWR (b) ATR

Figure 7: Reconstruction Error

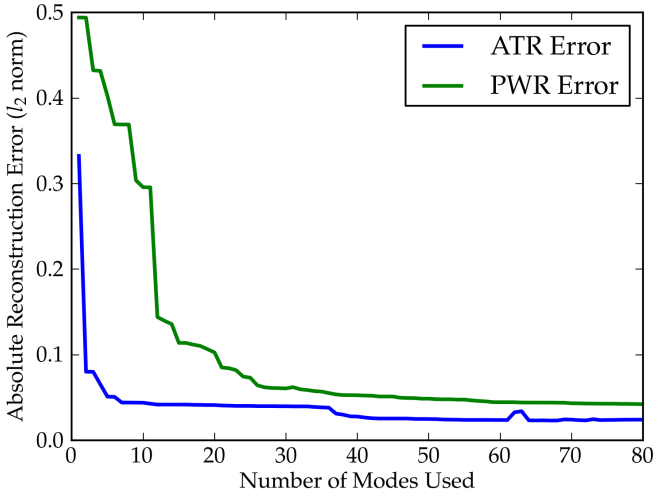


Figure 8: Reconstruction Error by Eigenmode Count

	$\bar{k}_1$	$\sigma_{\bar{k}_1}$	$\bar{k}_2$	$\sigma_{\bar{k}_2}$
KCODE	0.995077	0.000023	N/A	N/A
$50 \times 50 \times 50$ mesh	0.995017	0.000021	0.900928	0.000033
$25 \times 25 \times 25$ mesh	0.995011	0.000021	0.898198	0.000033
$10 \times 10 \times 10$ mesh	0.994977	0.000021	0.879747	0.000036
$5 \times 5 \times 5$ mesh	0.994924	0.000021	0.831998	0.000042

Table 5: Statistics of Runs

such, storing the squares of the tallies necessary to do error propagation in MCNP is not implemented. As such, the only way to measure statistics is to repeatedly run the same problem over and over again with different starting seeds until there are enough datapoints to calculate the variance of the sample.

To see if resolution has any impact on statistics, the ATR run was modified to use a $50 \times 50 \times 50$ mesh for the fission matrix, generating a 125000×125000 fission matrix. 25 runs were done and the mesh was aggregated into smaller meshes of size $25 \times 25 \times 25$, $10 \times 10 \times 10$, and $5 \times 5 \times 5$. The average of the resulting values along with the standard deviation of those values is summarized in Table 5.

It is worth noting that the statistics do get worse with decreasing mesh size, but, for example, the $5 \times 5 \times 5$ k_2 is not equal to the $50 \times 50 \times 50$ k_2, even within statistical variation. It appears that there is a significant bias effect for the non-fundamental modes at play with regards to mesh size. The same can be said of the $50 \times 50 \times 50$ k_1 value as compared to the KCODE k_1, but not to the same extreme. σ_{k_1} for all fission matrix runs was smaller than KCODE's, likely owing to the greater number of cycles contributing to the final result.

Another interesting test was to find the statistics on the first 100 eigenvalues for the $50 \times 50 \times 50$ run. This turned up a surprising result shown in Figure 9. Further exploration showed that mode 27 from run 20 had some regions inside of it with abnormally large values, as shown in Figure 10. The average mode is the one with run 20 removed but the other 24 combined. Removing all values above a certain threshold, the run 20 mode 27 is once again similar to the average, as shown in Figure 11, implying that the small number of very large values that caused the shift are likely statistical noise. Further, removing mode 20 completely from all calculations yields the statistics shown in Figure 12.

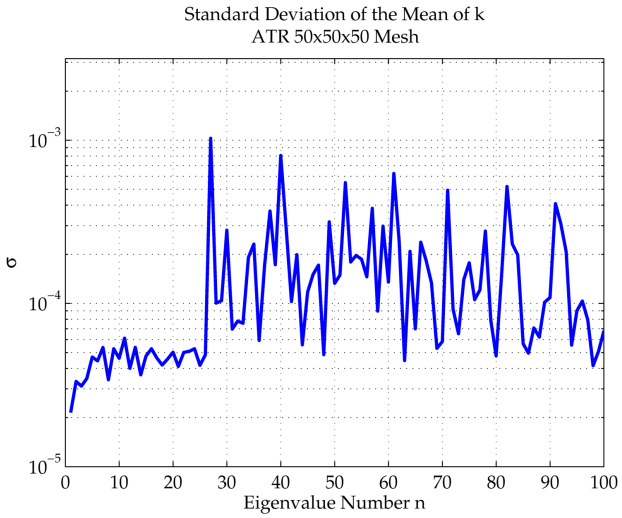


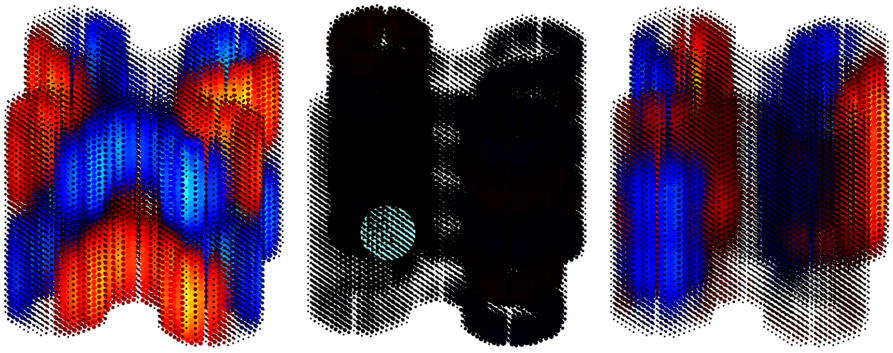
Figure 9: Standard Deviation by Eigenmode

5 Future Work

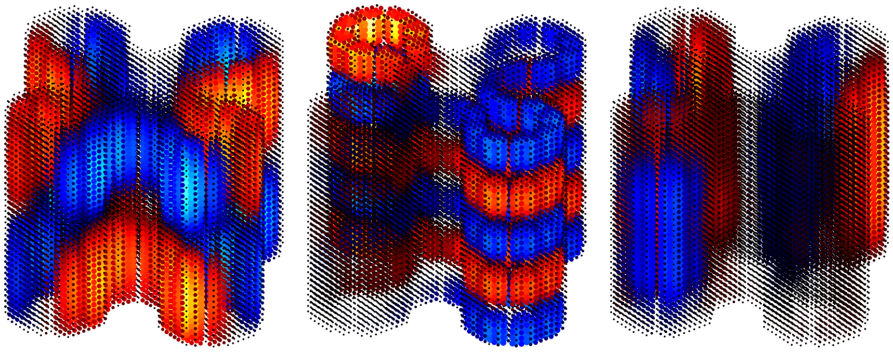
During the course of this work, many avenues for improvement and future investigation were uncovered. Many of these avenues were not pursued due to time constraints, so they are presented here as recommendations for continuations of this work by future groups.

First, improvements are possible in the algorithms themselves. The algorithms in ARPACK as it is typically implemented are not parallel. Since most of the time is spent doing matrix-vector math, parallelizing at least that component of the calculation is in principle fairly straightforward. This would be a benefit especially for extremely large matrices, where it can still take a large amount of time to solve for any useful number of eigenvectors.

Another ongoing point of concern is the existance of imaginary components that occasionally appear in the eigenvalues of the fission matrix. This phenomenon was observed during the course of this project, but not systematically investigated. It is investigated in [1], which makes the preliminary conclusion that the imaginary components appear to be solely due to statistical variation. However, a more thorough investigation of these components with a larger parameter space would be necessary in order to gather convincing evidence of this suspicion.

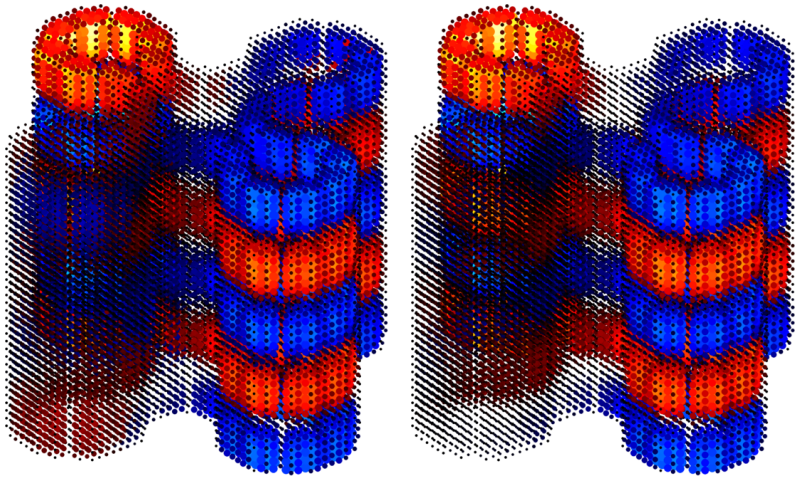


(a) Run 20 - Mode 26 (b) Run 20 - Mode 27 (c) Run 20 - Mode 28



(d) Average - Mode 26 (e) Average - Mode 27 (f) Average - Mode 28

Figure 10: Mode Comparison in 3D, Run 20 vs. Average



(a) Run 20 - Mode 27 - Modified (b) Average - Mode 27

Figure 11: Mode Comparison in 3D, Run 20 vs. Average, With Abnormal Values Removed

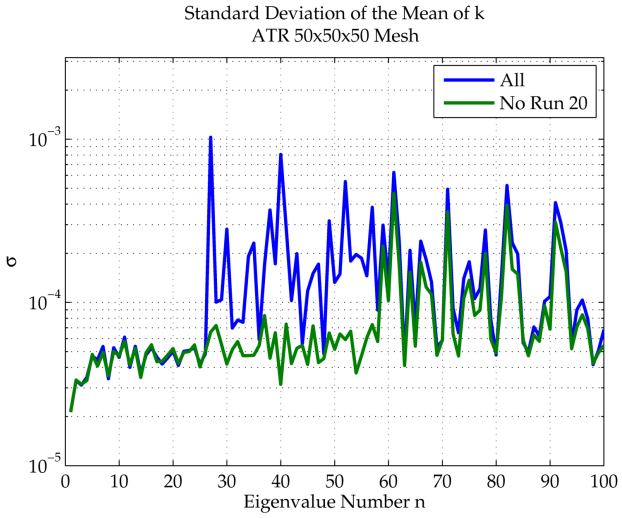


Figure 12: Standard Deviation With and Without Run 20

Finally, only a limited set of perturbations were explored in this project, and more interesting ones may be possible. For example, one might investigate the transitions from a core in cold-clean conditions to hot-clean conditions. It might also be interesting to investigate other temperature effects, void effects, or xenon oscillations. In principle, any type of transition can be investigated providing MCNP has the capability to simulate the perturbed case.

6 Conclusion

Fundamentally, most of the groundwork for using the fission matrices that MCNP produces has been completed. It is relatively cost-effective to generate fission matrices with meshes of acceptably fine resolution, with the primary limiters being memory and storage space. Large matrices provide more accurate results as long as the statistics are sufficient, and if they are not, the matrix can be reduced in size until they are sufficient. By switching from the power method to the implicitly restarted Arnoldi method, speedups on the order of 1500 fold were attained, proving this an effective algorithm for handling these large matrices. The calculated eigenvalues converged towards the `KCODE` result with expanding matrix size, and also appeared to have superior statistical properties due to the reduced number of discarded cycles, with the notable caveat of a bias appearing for insufficiently fine mesh resolutions. Lastly, the fundamental mode can be reconstructed in a perturbed space with relatively small errors with few eigenmodes used.

7 Acknowledgements

We would like to gratefully acknowledge the guidance of our mentor, Forrest Brown, and everyone else at Los Alamos National Laboratory who made the Computational Physics Worshop and this project possible, especially Scott Runnels, the organizer of the workshop.

References

[1] Forrest B. Brown, Sean E. Carney, Brian C. Kiedrowski, and William R. Martin. Fission matrix capability for MCNP, part I -

Theory. Technical Report LA-UR-13-20429, Los Alamos National Laboratory, May 2013. URL https://laws.lanl.gov/vhosts/mcnp.lanl.gov/pdf_files/la-ur-13-20429.pdf. Intended for Mathematics & Computation 2013, Sun Valley, ID, USA.

[2] David S Watkins. Understanding the QR algorithm, part II. URL http://www.researchgate.net/publication/228966308_Understanding_the_QR_algorithm_Part_II/file/79e4150e24b0e6b291.pdf.

[3] Y. Saad. *Numerical Methods for Large Eigenvalue Problems-classics edition*. SIAM, Philadelpha, PA, 2011. doi: 10.1137/1.9781611970739. URL http://dx.doi.org/10.1137/1.9781611970739.

[4] James S Warsa, Todd A Wareing, Jim E Morel, John M McGhee, and Richard B Lehoucq. Krylov subspace iterations for deterministic k-eigenvalue calculations. *Nuclear Science and Engineering*, 147(1):26–42, 2004.

[5] Bernhard Beckermann and Stefan Güttel. Superlinear convergence of the rational Arnoldi method for the approximation of matrix functions. *Numerische Mathematik*, 121(2):205–236, 2012.

[6] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide*, October 1997. URL http://www.caam.rice.edu/software/ARPACK/.

[7] Danny Sorensen. Caam 551: Advanced numerical linear algebra, matlab code, 2011. URL http://www.caam.rice.edu/~caam551/MatlabCode/matlabcode.html.

A Biorthogonality Relation

For an $N \times N$ matrix A with N distinct eigenvalues and a set of right eigenvectors $\{\mathbf{r}_i\}$ and associated left eigenvectors $\{\mathbf{l}_i\}$ (here written as column vectors), the following relation holds:

$$\begin{aligned} \langle A\mathbf{r}_j, \mathbf{l}_i \rangle &= \langle \lambda_j \mathbf{r}_j, \mathbf{l}_i \rangle \\ = \langle \mathbf{r}_j, A^* \mathbf{l}_i \rangle &= \langle \mathbf{r}_j, \lambda_i^* \mathbf{l}_i \rangle \end{aligned} \tag{A.1}$$

so

$$\left\langle \mathbf{r}_j, \mathbf{l}_i \right\rangle (\lambda_i - \lambda_j) = 0 \tag{A.2}$$

by sesquilinearity of the dot product defined in Section 2.1. This means that for $\lambda_i \neq \lambda_j$, the product $\langle \mathbf{r}_j, \mathbf{l}_i \rangle = 0$. Therefore, since the matrix A and its complex conjugate transpose A^* have sets of eigenvalues that are complex conjugates of each other, the sets of left and right eigenvectors $\{\mathbf{l}_i\}$ and $\{\mathbf{r}_j\}$ form a biorthogonal system.

In the more general case in which A has repeated eigenvalues, which is often observed to be the case with fission matrices, a more general relation is needed which makes use of the Jordan normal form. Any square matrix can be decomposed in the form $A = RJR^{-1}$, where J is a block-diagonal matrix where the individual blocks are Jordan blocks, which have an eigenvalue on the diagonal and ones on the superdiagonal. The number of Jordan blocks corresponding to an eigenvalue λ_i is equal to its geometric multiplicity, or the number of linearly independent eigenvectors corresponding to that eigenvalue ($= \dim \text{Null}(A - \lambda_i I)$). All semi-simple eigenvalues, i.e. those that have a geometric multiplicity equal to their algebraic multiplicty (their multiplicity as roots of the characteristic polynomial of A), have Jordan blocks of size 1. For more details, see [3, pp. 14-15] or any standard textbook on linear algebra. For the purposes of illustration, a Jordan block of the eigenvalue λ_i has the form:

$$\begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}$$

Writing the above decomposition in the form $AR = RJ$, it can be seen that R contains all the right eigenvectors of A. The set of indices of the columns in R that comprise the eigenvectors of the matrix A is the set of (column) indices in J where each new Jordan block starts; this is the set of indices at which a column vector of R is scaled by the corresponding λ_i through the matrix multiplication.

Alternatively, the decomposition can be written $R^{-1}A = JR^{-1}$, or, letting $L = R^{-1}$, $LA = JL$. This form reveals that L contains all the *left* eigenvectors of A in its rows. This time, however, the eigenvectors occur at a different set of (row) indices: By inspecting the properties

of left-multiplication of a matrix by a Jordan block, one can see that the true eigenvectors occur at the set of indices where each Jordan block *ends*; these are the indices at which a row vector in L is scaled by the corresponding λ_i in the matrix multiplication.

Since $LR = I$, the identity matrix, the rows $\{\mathbf{l}_i\}_{i=1}^N$ and the columns $\{\mathbf{r}_i\}_{i=1}^N$ form a biorthogonal basis of $\mathbb{C}^N$. From this fact, it is possible to make a restricted conclusion regarding the biorthogonality of eigenvectors of a matrix: The sets of left and right eigenvectors of a matrix, with those vectors deleted that do not correspond to semi-simple eigenvalues, form a biorthogonal system. This is because, for semi-simple eigenvalues, the corresponding left and right eigenvectors have the same row indices in L as column indices in R, since Jordan blocks corresponding to semi-simple eigenvalues always have size 1.

This manipulation of the Jordan normal form offers another insight in the case where the fission matrix $\bar{F}$ does not have a complete, linearly independent set of eigenvectors: One can augment the existing eigenvectors with principal vectors from the Jordan normal form, forming two complete biorthogonal bases of $\mathbb{C}^N$ in which any arbitrary source distribution can be written as $\mathbf{s} = \sum_{i=1}^N a_i \mathbf{l}_i$, for example (see Section 2.1). More care is required, however, in extracting the j-th transition coefficient via $a_j = \langle \mathbf{s}, \mathbf{r}_j \rangle$. The index j is an index into a column of R, which includes principal vectors from the Jordan normal form. Taking into account the above-mentioned indexing irregularities, this means that while $\mathbf{l}_j$ may be an eigenvector, this does not imply that $\mathbf{r}_j$ is as well, and vice versa.

Modeling X-ray Thomson scattering spectra of warm dense matter

(Didier Saumon and Charles Starrett, mentors)

LA-UR-13-26717
Approved for public release; distribution is unlimited.

Title: Modeling X-ray Thomson scattering spectra of warm dense matter

Author(s): Perkins, David J.
Souza, Andre N.
Saumon, Didier
Starrett, Charles E.

Intended for: Report
Web

Issued: 2013-08-26

Modeling X-Ray Thomson Scattering Spectra of Warm Dense Matter

David Perkins
University of California, Los Angeles
Department of Physics and Astronomy

Andre Souza *
University of Michigan, Ann Arbor
Department of Mathematics

Mentors: **Didier Saumon** and **Charles Starrett**
Los Alamos National Laboratory, XCP-5 †

Contents

*Partially supported by the Emeritus Professor Maxwell Reade Fund.
†Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396

1 Introduction

Warm dense matter (WDM) is sometimes described as a "solid density plasma" where neither weakly coupled plasma theory nor condensed matter theory are applicable. More specifically, WDM has densities of $\sim 0.1 - 50$ times solid density, temperatures of $\sim 1 - 100$ eV, is partially ionized, globally neutral, has strongly correlated particles (due to interactions), and partial electron degeneracy. In nature, WDM may be found in the interiors of planets such as Jupiter or in the envelopes of white dwarf stars, and in man-made processes such as inertial confinement fusion (ICF), WDM exists as a transient state of matter. To model these physical systems, the equation of state as well as opacity, conductivity, and diffusion coefficients must be known.

Experimentally WDM is probed with an experimental technique known as X-Ray Thomson Scattering (XRTS). In such an experiment X-rays photons are scattered off of WDM and the resulting photons are measured at a given solid angle. The resulting spectrum gives information on the density and temperature of WDM, but a model of WDM and the scattering experiment is often required to interpret the results. As a result, these experiments also serve as a method of validation for potential models of WDM.

The work presented here focuses on the model of WDM proposed by Starrett and Saumon [1] and a model for scattering based off of Chihara's work [2]. The model proposed by Starrett and Saumon is an average-atom model that is coupled with a two component plasma model, hereafter referred to as an AA+TCP model. The AA+TCP model is an ab initio model with no free parameters, only requiring knowledge of the element under investigation, the density, and the temperature. The outputs from the model completely determine all the input parameters needed for Chihara's structure function (chemical potential, elastic feature, screening potential, bound-state wavefunctions, and free wavefunctions), which in turn determines the double differential scattering cross-section of the experiment. This approach to determining the outcome of an XRTS experiment parallels that of Johnson, et al. [3], but expands the work in many respects. For example, the AA+TCP model completely determines all the input parameters to the structure function; hence no ad hoc approximations are necessary to close the system. Also, the free-electron feature and the bound-free feature are modified to include collisions and occupation of free electron states, respectively.

2 The Structure Function

In order to compare with experiment, knowledge of the double differential cross-section is necessary. The double differential cross-section is related to the structure function via the following formula

$$\frac{d\sigma}{d\omega_1 d\Omega} = \left(\frac{d\sigma}{d\Omega}\right)_{\text{Th}} \frac{\omega_1}{\omega_0} S(k,\omega), \tag{1}$$

where $\frac{d\sigma}{d\omega_1 d\Omega}$ is the double differential cross-section, $\left(\frac{d\sigma}{d\Omega}\right)_{\text{Th}}$ is the classical Thomson scattering cross-section, ω_0 is the incoming photon energy, ω_1 is the outgoing photon energy, $\omega = \omega_0 - \omega_1$, k is the change in photon wavenumber given by the formula $k = 2k_0 \sin(\theta/2)$ where k_0 is the initial photon wavenumber and θ is the scattering angle, and S is the structure function. According to Chihara's formula [2] the structure function may be decomposed into three terms:

$$S(k,\omega) = \underbrace{S_{ii}(k,\omega)}_{\text{Elastic Feature}} + \underbrace{S_{ee}(k,\omega)}_{\text{Free-electron Feature}} + \underbrace{S_b(k,\omega)}_{\text{Bound-free Feature}} . \tag{2}$$

The elastic feature (also called ion-ion feature) represents scattering off of electrons that follow the ion motion, the free-electron feature (also called electron-electron feature) represents scattering off of free electrons, and the bound-free feature arises as photons ionize electron to a free state during the scattering process. In the sections that follow, detailed explanations of each are given.

2.1 Elastic Feature

The elastic scattering structure function $S_{ii}(k,\omega)$ results from photons scattering elastically off of electrons that follow ion motion within the plasma. This includes both bound electrons and screening electrons. The term takes the following form [3]:

$$S_{ii}(k,\omega) = |f(k)+q(k)|^2 S_{ii}(k)\delta(\omega). \tag{3}$$

Here, $f(k)$ and $q(k)$ are the Fourier transforms of the bound and screening electron densities, respectively, due to a single ion. The function $S_{ii}(k)$ is the static structure factor, which describes the spatial structure of ions in the matter. The term $\delta(\omega)$ is the Dirac delta function, centered at $\omega = 0$, an approximation reflecting the fact that photon energy is unchanged in an elastic scattering event.

In practice, $S_{ii}(k,\omega)$ is very straightforward to compute. As is clear from Eq. (3), it simply consists of computing the k-dependent part of the elastic feature $G(k) = |f(k)+q(k)|^2 S_{ii}(k)$ for a fixed k. The AA+TCP code computes $f(k)$, $q(k)$, and $S_{ii}(k)$ on a substantial range of values for k. Given them, it is a trivial matter to compute $G(k)$ on the same k array. An example of $S_{ii}(k)$, along with the resultant function $G(k)$, is plotted in Fig. 1. The peak in $G(k)$ corresponds with the peak in $S_{ii}(k)$ and falls off at large k with the electron density terms as $S_{ii}(k) \approx 1$. With $G(k)$ computed on the k grid given by the AA+TCP code, the value of $G(k)$ is easily found for any fixed k by cubic spline interpolation.

Of course, the delta function is only an approximation and the actual feature has some nonzero extent in frequency space. In fact, the elastic scattering feature width due to doppler shifts resulting from thermal ion motion is approximated by

$$\Delta\omega = \omega_0\sqrt{(2\ln 2)k_BT/Mc^2},$$

where k_B is the Boltzmann constant, T is the temperature, M is the ion mass, and c is the speed of light. In typical WDM applications this width is on the order of 10^{-1} eV. This is minuscule compared with all other features in the structure function as well as with the resolving power of current experimental apparatus. It is thus safe to approximate the feature as a delta function.

2.2 Free-electron Feature

The formula for the free-electron feature (in atomic units) is given by [3]

$$S^0_{ee}(k,\omega) = -\frac{1}{1-\exp(-\omega/k_BT)}\frac{k^2}{4\pi^2 n_e}\mathrm{Im}\left[\frac{1}{\varepsilon(k,\omega)}\right], \tag{4}$$

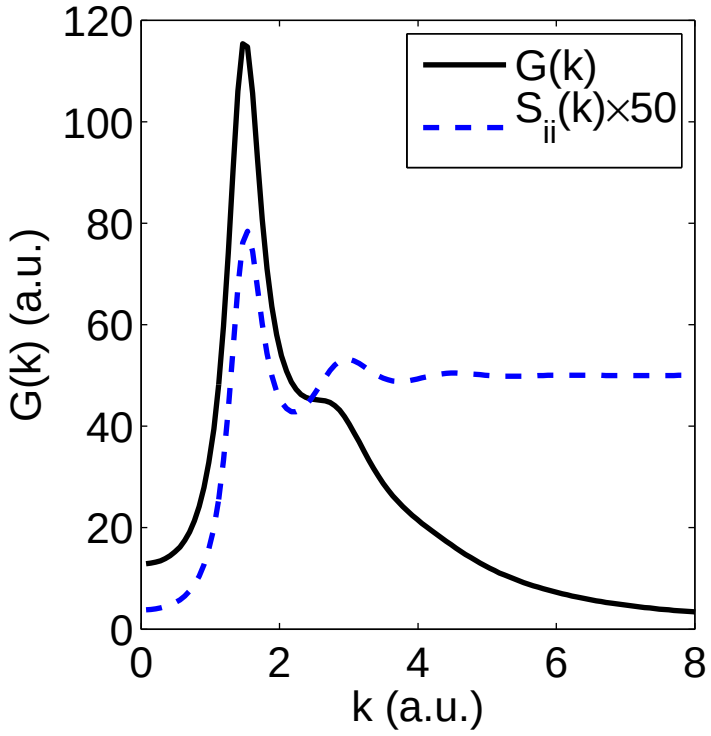
where $\omega = \omega_0 - \omega_1$ is the change in photon energy, k_BT is the electron temperature in units of energy, k is the change in photon momentum, n_e is the number density of electrons, and $\varepsilon(k,\omega)$ is the dielectric function. S^0_{ee} is related to S_{ee} by $S_{ee}(k,\omega) = Z_f S^0_{ee}(k,\omega)$ where Z_f is the number of free electrons per ion. It should be noted that the AA+TCP model has two different Z_f that may be used, one associated with AA and another associated with TCP. For this work, the AA Z_f was used for all results.

In order to calculate the free-electron feature, an approximation to the dielectric function known as the random phase approximation (RPA) is used. The RPA dielectric function may be calculated via the following formula [3]

$$\varepsilon(k,\omega) = 1 + \frac{4}{\pi k^2}\int_0^\infty \mathcal{F}(p^2/2;\mu_e,k_BT)p^2\left(\int_{-1}^1\left[\frac{1}{k^2-2pk\mu+2\omega+i\nu}+\frac{1}{k^2-2pk\mu-2\omega-i\nu}\right]d\mu\right)dp \tag{5}$$

$$\mathcal{F}(p^2/2;\mu_e,k_BT) = \frac{1}{1+\exp[(p^2/2-\mu_e)/k_BT]}. \tag{6}$$

Figure 1: The static structure factor $S_{ii}(k)$ and the k-dependent part of the elastic feature $G(k) = |f(k) + q(k)|^2 S_{ii}(k)$ for aluminum at $k_B T = 1$ eV and solid density.



The semicolon notation is a way of showcasing a parameter dependence. Note that $\mathcal{F}$ is a Fermi occupation factor, μ_e is the chemical potential, and ν is a quantity that will eventually be taken to zero[1]. The details on how to compute the RPA structure function from Eq. (5) are given in §A. Note that there is a removable discontinuity at $\omega = 0$. This scenario is handled for the RPA case in §A.3.

In Fig. 2 an example of a typical free-electron feature is shown. The two figures are manifestations of two broad S_{ee} regimes: the left is in the collective regime while the figure on the right is in the non-collective regime. The collective regime probes bulk plasma properties while the non-collective regime probes individual atoms. A simple number known as the dimensionless scattering parameter (usually denoted by the symbol α) may be attached to these scenarios. It is defined by the formula (in atomic units) [5],

$$\alpha \equiv \frac{k_s}{k}, \tag{7}$$

where

$$k_s = \sqrt{\frac{k_B T}{4\pi n_e} \frac{F_{1/2}(\mu_e/k_B T)}{F_{-1/2}(\mu_e/k_B T)}}$$

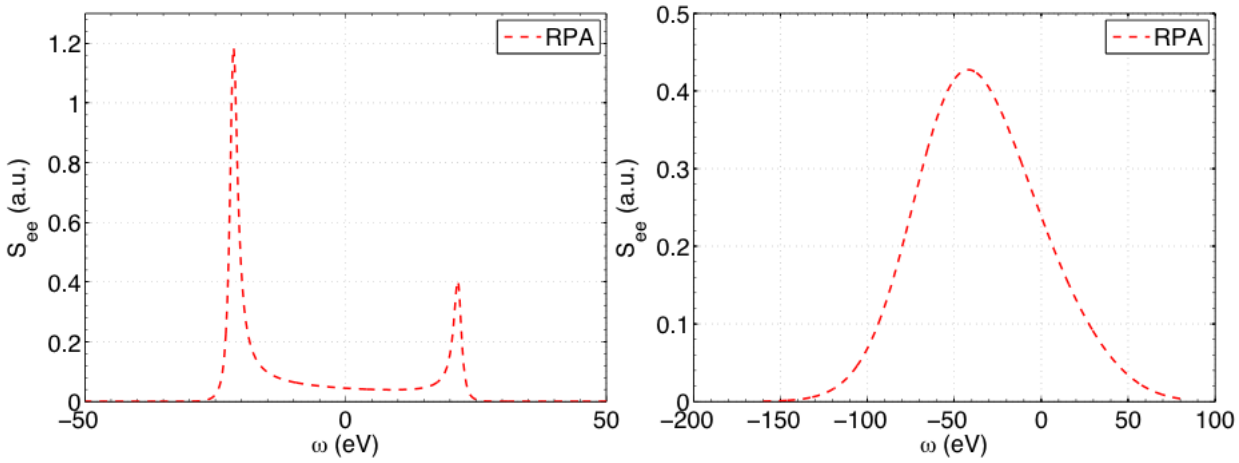
is the electron screening wavenumber,

$$F_\nu(y) = \frac{1}{\Gamma(\nu+1)} \int_0^\infty \frac{x^\nu}{\exp(x-y)+1} dx,$$

is the complete Fermi-Dirac integral, and Γ is the gamma function. The dimensionless scattering parameter is a measure of the WDM regime being probed. If $\alpha > 1$, then S_{ee} is in the collective regime and if $\alpha < 1$ then S_{ee} is in the non-collective regime.

[1] This represents a "collisionless" approximation and is the essence of the RPA approximation

Figure 2: The free-electron feature for beryllium at 20 eV and solid density for an initial photon energy of $\omega_0 = 2960$ eV at scattering angles of 20° (left) and 130° (right). The parameter α for 20° is $\alpha = 2.480$ and for 130°, $\alpha = 0.475$.



As a consequence of the detailed balance relation [3], which is given by

$$S(k,\omega) = e^{\omega/k_B T} S(k,-\omega),$$

the ratio of the peak heights provides a way of measuring the temperature in warm dense plasma. The peaks are called plasmon resonances.

2.3 Beyond the RPA

As mentioned previously, the RPA does not take into account collisions in WDM. From now on, the RPA dielectric function will be denoted by $\varepsilon^{\mathrm{RPA}}$. A way to extend the dielectric function to include more physics is with the extended Mermin ansatz [5] which results in the formula:

$$\varepsilon^M(k,\omega) = 1 + \frac{[1+i\nu(\omega)/\omega][\varepsilon^{\mathrm{RPA}}(k,\omega+i\nu(\omega))-1]}{1+i[\nu(\omega)/\omega][\varepsilon^{\mathrm{RPA}}(k,\omega+i\nu(\omega)-1]/[\varepsilon^{\mathrm{RPA}}(k,0)-1]}.$$

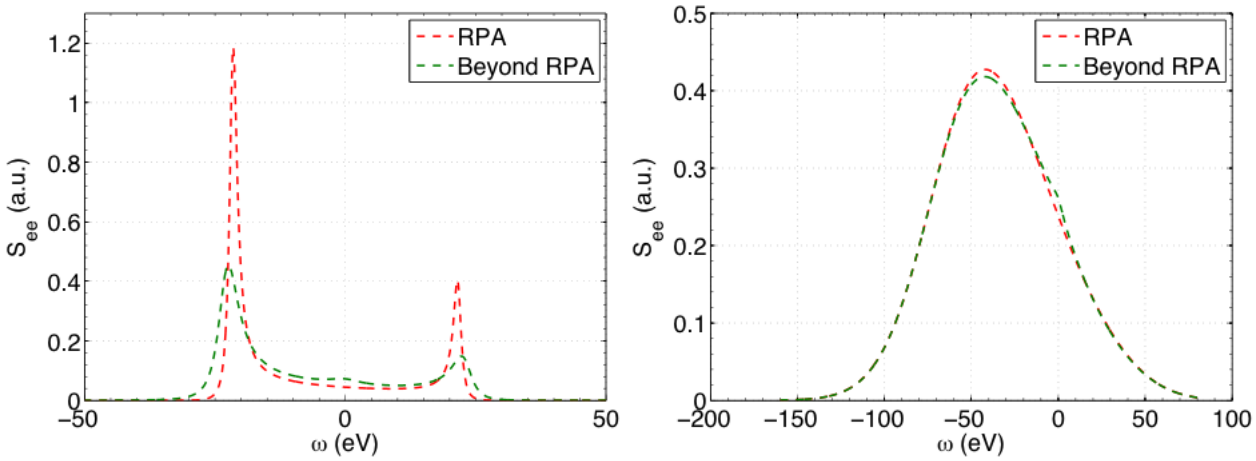
The introduction of the complex frequency $\omega + i\nu$ accounts for damping by collisions. Furthermore the dynamic collision frequency ν must be approximated. With the Born approximation [5] ν^B (in atomic units) is given by

$$\nu^B(\omega) = -i\frac{\Omega_0^2}{24\pi^3 Z_f}\int_0^\infty k^6 [V_{ei}^S(k)]^2 S_{ii}(k)\frac{1}{\omega}\left[\varepsilon_e^{\mathrm{RPA}}(k,\omega) - \varepsilon_e^{\mathrm{RPA}}(k,0)\right] dk,$$

where Ω_0 is a normalization volume, V_{ei}^S is the potential from the AA+TCP model, and S_{ii} is the ion-ion structure function (elastic feature). The superscript B on the collision frequency denotes the fact that it comes from the Born approximation. It is difficult to find any information on what and where the quantity Ω_0 comes from. Since integral formulas are used for quantities in this work (as opposed to summations), it seems to be the case that $\Omega_0 = 1$. This will henceforth be assumed.

In Fig. 3 an example of a typical free-electron feature is shown. The details on how to compute the beyond RPA structure function are given in the appendix. The figure on the left is in the collective regime while the figure on the right is in the non-collective regime. One can see the effect that the

Figure 3: The free-electron feature for beryllium at 20 eV and solid density for an initial photon energy of $\omega_0 = 2960$ eV at scattering angles of 20° (left) and 130° (right). The red dashed line is the RPA free-electron structure function while the green dashed line is free-electron structure function computed with the Born-Mermin ansatz.



dynamic collision frequency has on the free-electron structure function. In the collective regime the structure function gets damped and shifted outwards while in the non-collective regime there is not much change. These two scenarios are typical for the free-electron feature. Sometimes in the non-collective regime things get shifted to the left or right and damped more than what is shown in the figure, but in general the structure function resembles the RPA free-electron feature. The astute reader may notice a bump at $\omega = 0$ in both the collective and non-collective regimes. This seems to be a consequence of the model and not a numerical resolution error.

2.4 Bound-free Feature

The bound-free feature $S_b(k,\omega)$ arises as X-ray photons ionize a bound electron to a free state during the scattering process. By conservation of energy $S_b(k,\omega)$ is nonzero for energy transfers from the photon sufficiently large to ionize the electron.

The formula for the bound-free structure function in atomic units is (see [3])

$$S_b(k,\omega) = \frac{1}{8\pi^3} \sum_{n\ell m} \frac{o_{n\ell}}{2\ell+1} \int p \left| \langle \psi_p | \, e^{i\mathbf{k}\cdot\mathbf{r}} \, | \psi_{n\ell m} \rangle \right|^2 \mathrm{d}\Omega_p, \tag{8}$$

where n, ℓ, and m are the electron bound state quantum numbers, p is the magnitude of the momentum of the final electron state, ψ_p is the final (free) electron wavefunction, $\psi_{n\ell m}$ is the initial (bound) electron wavefunction, $e^{i\mathbf{k}\cdot\mathbf{r}}$ is the final photon state, and the differential $\mathrm{d}\Omega_p$ implies integration over all angles of the final electron momentum. The occupation number $o_{n\ell}$ is the average number of electrons occupying the $n\ell$ bound state, and is given by

$$o_{n\ell} = 2(2\ell+1)\mathcal{F}(\epsilon_{n\ell}),$$

where $\mathcal{F}(\epsilon_{n\ell})$ is the Fermi occupation factor defined in Eq. (6) (the other arguments in Eq. (6), being material and state properties, are implied hereafter). The energy $\epsilon_{n\ell}$ is the initial bound electron energy, which is independent of m. The final electron state will have energy

$$\epsilon_p = \omega + \epsilon_{n\ell},$$

and the magnitude of the final momentum is

$$p = \sqrt{2\epsilon_p}.$$

According to Johnson, et al. [3], when assuming average-atom final electron states the formula in Eq. (8) can be reduced to the much more computationally tractable formula

$$S_b(k,\omega) = \sum_{n\ell} \frac{2p}{\pi} o_{n\ell} \sum_{\ell_1\ell_2} A_{\ell_1\ell\ell_2} \left| I_{\ell_1\ell\ell_2}(p,k) \right|^2 . \tag{9}$$

In Eq. (9), ℓ_1 is the orbital angular momentum quantum number of the final electron state and ℓ_2 is the orbital angular momentum of the final photon state. The factor $A_{\ell_1\ell\ell_2}$ is given by

$$A_{\ell_1\ell\ell_2} = (2\ell_1+1)(2\ell_2+1) \begin{pmatrix} \ell_1 & \ell & \ell_2 \\ 0 & 0 & 0 \end{pmatrix}^2 , \tag{10}$$

where $\begin{pmatrix} \ell_1 & \ell & \ell_2 \\ 0 & 0 & 0 \end{pmatrix}$ is the Wigner 3-J symbol that arises from integration over the spherical harmonics that form the angular portion of the wavefunctions in the matrix element in Eq. (8). The function $I_{\ell_1\ell\ell_2}(p,k)$ is given by

$$I_{\ell_1\ell\ell_2}(p,k) = \frac{1}{p} e^{i\delta_{\ell_1}(p)} \int_0^\infty P_{\epsilon\ell_1}(r) j_{\ell_2}(kr) P_{n\ell}(r) \mathrm{d}r, \tag{11}$$

where the P functions are the radial portions of the corresponding electron wavefunctions, scaled by r for ease of spherical integration (i.e. $P_{n\ell}(r) = r\psi_{n\ell}(r)$, where $\psi_{n\ell}(r)$ is the radial portion of $\psi_{n\ell m}(\mathbf{r})$), the subscript $\epsilon\ell_1$ refers to the free state with energy ϵ_p and orbital angular momentum quantum number ℓ_1, and j_{ℓ_2} is the spherical Bessel function indexed by ℓ_2 and represents the radial part of the photon final state. The value of $\delta_{\ell_1}(p)$, the phase of the final electron state, is inconsequential because only the norm of $I_{\ell_1\ell\ell_2}(p,k)$ is used in the computation of $S_b(k,\omega)$.

Some corrections to Eqs. (9) and (11) are necessary. First, while Eq. (9) accounts for the occupation of initial (bound) electron states through the occupation number $o_{n\ell}$, it fails to fully consider the occupation factor of free-electron states. This is given by $\mathcal{F}(\epsilon_p)$ according to Eq. (6). Since a bound electron can only be ionized to a previously unoccupied free state, Eq. (9) must be corrected by a factor of $1-\mathcal{F}(\epsilon_p)$ to reflect the likelihood that the final state to which the electron is ionized is unoccupied.

The other necessary corrections consider weakly bound states. The AA+TCP model that supplies the wavefunctions used in Eq. (9) includes corrections to weakly bound states to ensure a smooth transition of bound states into the continuum and to truncate the weakly bound wavefunctions[1]. There are two specific corrections: the first is a cutting function $f_{\mathrm{cut}}(r)$ that multiplies the square of the bound state wavefunction and the second is a weighting factor $M(\epsilon_{n\ell})$. Details on the specific formulation of these corrections can be found in [1].

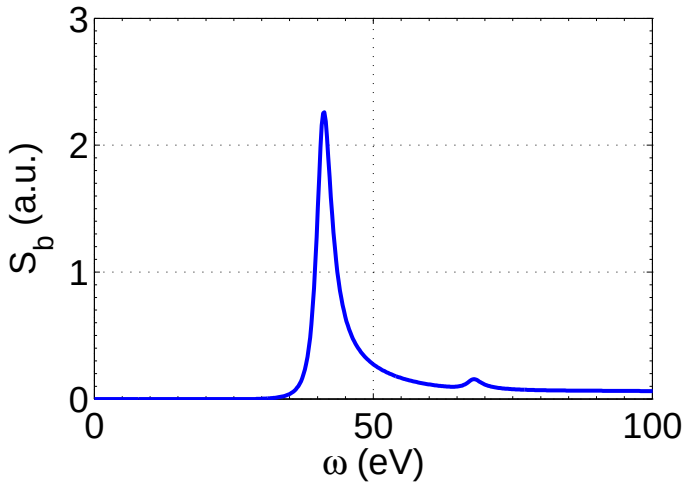
With these corrections, Eqs. (9) and (11) become

$$S_b(k,\omega) = \sum_{n\ell} \frac{2p}{\pi} o_{n\ell} \mathcal{F}(\epsilon_{n\ell}) M(\epsilon_{n\ell}) \sum_{\ell_1\ell_2} A_{\ell_1\ell\ell_2} \left| I_{\ell_1\ell\ell_2}(p,k) \right|^2 \tag{12}$$

$$I_{\ell_1\ell\ell_2}(p,k) = \frac{1}{p} e^{i\delta_{\ell_1}(p)} \int_0^\infty P_{\epsilon\ell_1}(r) j_{\ell_2}(kr) P_{n\ell}(r) \sqrt{f_{\mathrm{cut}}(r)} \mathrm{d}r. \tag{13}$$

Details on the computational implementation of these formulas can be found in Appendix B. Briefly, the computational method is a convergence test that consists of computing Eq. (12) using Eqs. (10) and (13) up to a maximum value of ℓ_1 and ℓ_2 and repeating the computation for successively larger maximum values of ℓ_1 and ℓ_2 until successive results differ by a negligible amount.

Figure 4: The bound-free feature for chromium at $k_BT = 5$ eV, density $\rho = 7.19\ g/cm^3$, an initial photon energy of 4750 eV, and scattering angle $\theta = 40$ degrees. The large narrow feature is a result of electrons ionized from the 3p state, while the smaller bump results from the 3s state.



An example of a bound-free structure function S_b is pictured in Fig.4 as a function of ω for a fixed k. This example is of chromium at temperature $T = 5$ eV, solid density ($\rho = 7.19$ g/cm^3), with incident X-ray photon energy $\omega_0 = 4750$ eV, and at a scattering angle of $\theta = 40°$. The structure function is zero up to a point, here around $\omega = 30$ eV, reflecting the fact that S_b is zero where ω is too small an energy transfer to ionize any bound electrons. Multiple peaks occur in the figure; these correspond with different bound states. The largest peak is results from electrons ionized from the 3p state, which is the most weakly bound state in chromium at this density and temperature and has an ionization energy of 28.9 eV. The smaller peak corresponds with the 3s state, whose ionization energy is 55.9 eV. This is characteristic of bound-free structure functions; the most prominent peak tends to arise from the most weakly bound electrons. Of course, there are plenty of other occupied bound electron states that contribute to S_b in this chromium case; however, these have such large ionization energies that they are far from the ω range pictured in the figure (the smallest is around 500 eV).

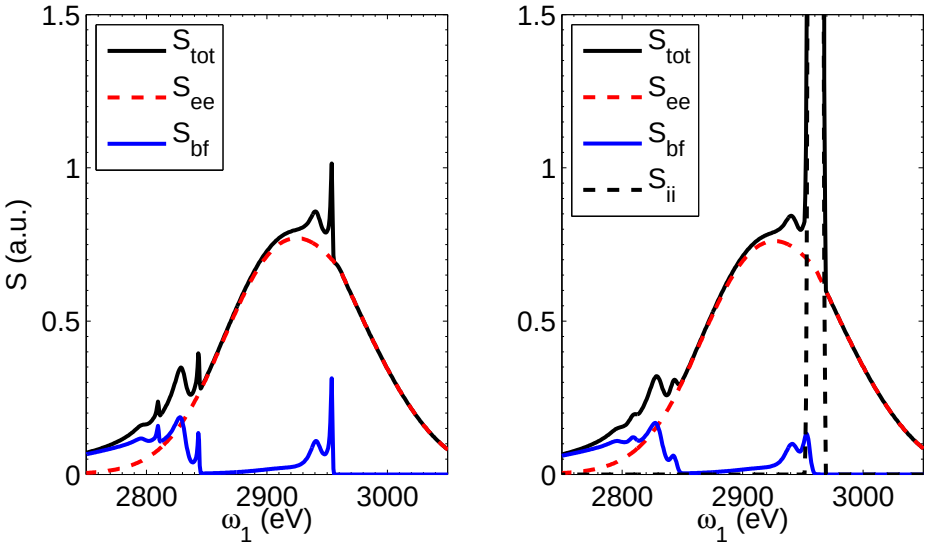
3 Results

In this section the theoretical spectra for various cases are computed. In §3.2.2 the free-electron feature is computed with the RPA; in all other cases the free-electron feature is computed with the Born-Mermin ansatz. A description of the process of preparing the theoretical structure function with experiment is given in §3.1, including a discussion of convolution of the theoretical scattering spectrum with a source spectrum. A limited verification study is presented in §3.2 comparing calculations described in this document to several published results. A preliminary exploration of parameter space is presented in §3.3, in which dependence of the structure function on temperature and scattering angle is explored.

3.1 Theoretical Spectra and Convolution

In §2 the structure function is presented and each of its constituent parts are explained in detail. The total structure function is, according to Eq. (2), the sum of the elastic, free electron, and bound-free terms. While useful for theoretical purposes, however, the total structure function alone is not sufficient

Figure 5: The theoretical spectrum for aluminum at a density of $\rho = 2.7$ g/cm^3 and $k_BT = 50$ eV for an initial photon energy of 2960 eV and a scattering angle of $\theta = 130°$. The unconvolved spectra is on the left and the convolved spectra is on the right. A 5 eV Gaussian was used for the convolution and S_{ii} makes its appearance as a bump in the figure on the right, whereas originally it was a delta function centered at 2960 eV and absent from the figure on the left. The height of the total structure function at 2960 eV is ~ 245 atomic units.



to produce an XRTS profile comparable to experiment. As is seen in Eq. (1), the structure function must be scaled by prefactors.

However, a more pressing concern is that any experimental incident X-ray source is distributed over a range of photon energies. As such, the spectrum resulting from a single photon energy cannot be observed experimentally. The most theoretically sound method of dealing with this complication would be to compute the spectrum for each incident photon energy in the range and then to combine them as an average weighted by the incident spectrum. However, this would be prohibitively expensive to compute. A more economical method consists of producing the scattering profile for a single photon energy (preferably the center of the source spectrum or the energy at the peak of the source spectrum) and then convolving the profile with the source spectrum. The viability of this approach depends on the scattering profile varying only minimally as the incident photon energy varies over the source spectrum range; in all cases observed thus far, this is indeed the case.

An example of a theoretical scattering profile for a single incident photon energy and the resulting convolved spectrum is pictured in Fig. 5. The profile was convolved with a normalized Gaussian with full width at half-maximum 5 eV. In simulating an experimental scattering spectrum it is possible to convolve the theoretical profile with the experimentally measured source spectrum or any reasonable profile for a source. The effects of convolution are clear; the most narrow features are the most drastically affected, sometimes washing out almost completely, while broad features remain mostly untouched. It is important to recognize the appearance of the large feature near $\omega_1 = \omega_0$ after convolution; this is the elastic scattering feature, which was expressed theoretically as a delta function (see §2.1) and thus takes on the shape of the source spectrum with which it is convolved. It should also be noted that the convolved profile in Fig. 5 is also scaled with the prefactors shown in Eq. (1). In this way the resultant scattering spectrum is ready for comparison with experimental data.

Panel	n_e (cm^{-3})	k_BT (eV)	ω_0 or λ_0	θ (degrees)	μ_e (a.u.)	Z_f
Top Left	7.5×10^{23}	12	6180 (eV)	25	0.930851	2
Top Right (Black)	1.0×10^{19}	200	532 (nm)	60	-105.475	2
Top Right (Red)	1.0×10^{19}	600	532 (nm)	60	-352.76	2
Top Right (Blue)	1.0×10^{19}	3000	532 (nm)	60	-2029.96	2
Bottom Left (Black)	1.0×10^{21}	0.5	4.13 (nm)	60	-0.0109079	2
Bottom Left (Red)	1.0×10^{21}	2	4.13 (nm)	60	-0.207043	2
Bottom Left (Blue)	1.0×10^{21}	8	4.13 (nm)	60	-1.44484	2
Bottom Right (Black)	1.0×10^{23}	0.8	0.26 (nm)	60	0.2862	2
Bottom Right (Red)	1.0×10^{23}	3	0.26 (nm)	60	0.247005	2
Bottom Right (Blue)	1.0×10^{23}	13	0.26 (nm)	60	-0.437541	2

Table 1: The parameters for the Fig. 6.

3.2 Comparison to Literature

One of the methods that was used to verify that calculations were being performed correctly was comparison with the existing literature where possible. Since the formulas used in this work are modifications of existing formulas and use WDM data given by the model of Starrett and Saumon [1] for the first time, perfect agreement is not expected in all cases. In this section a few of these comparisons are presented, including discussions of discrepancies where the models fail to agree. The primary sources of data with which to compare are [3], [4], and [5].

3.2.1 Free-electron Feature

Comparisons were made to Glenzer and Redmer[2] [5] to check that the results where being computed correctly for S_{ee}. In some cases there were noticeable differences between the computed values and what is shown in the article, but this may arise from poorly specified computational parameters in the published calculation. Increasing the number of grid points used for integration of the dynamic collision frequency and the RPA dielectric function did not change the computed results enough to account for the discrepancy. Thus it is unlikely that the difference is due to numerical inaccuracies in the calculation. In Table 1 the parameters for Fig. 6 are summarized. The parameter λ_0 is the initial photon wavelength. Compare Fig. 6 with figures 8 and 9 of [5]. Note that in atomic units the plasma frequency ω_{pl} is $\omega_{pl} = \sqrt{4\pi n_e}$. The discrepancy between figure 9 of [5] and the bottom left figure in Fig. 6 may come about due to the choice of $Z_f = 2$, while in [5] the value of this important parameter is not given.

3.2.2 Bound-free Feature

Johnson, Nilsen, and Cheng explore the bound-free feature for several cases in [3] and [4]. In Fig. 7 the bound-free feature computed as described in this report is shown for beryllium at $T = 20$ eV, $\rho = 1.85$ g/cm^3, $\omega_0 = 2960$ eV and two scattering angles: 30° and 150°. The single feature apparent in the profiles arises from the 1s state, the only bound electron state present for beryllium at the given temperature and density. This is an analogous figure to that presented in Fig. 5 of [3]. Some similarities are apparent: the general structure of the functions appears the same in each figure, as do the location of peaks and the fact that the peak height increases by about a factor of 10 from the 30° case to the 150° case. The most marked difference between Fig. 5 of [3] and Fig. 7 is that the peaks in

[2]Note that there is a typo in the formula for the Debye expression in [5]. The correct formula (in atomic units) is $\kappa_i^2 = Z_f^2 n_i 4\pi / k_B T$.

Figure 6: The free-electron feature $S(k,\omega)$ for various cases. In all cases the dashed line corresponds to the RPA structure function and the solid lines are the structure function computed with the extended Mermin Ansatz using the TCS model as described in [5]. The parameters used to make each figure are summarized in Table 1. These plots can be directly compared to figures 8 and 9 of [5].

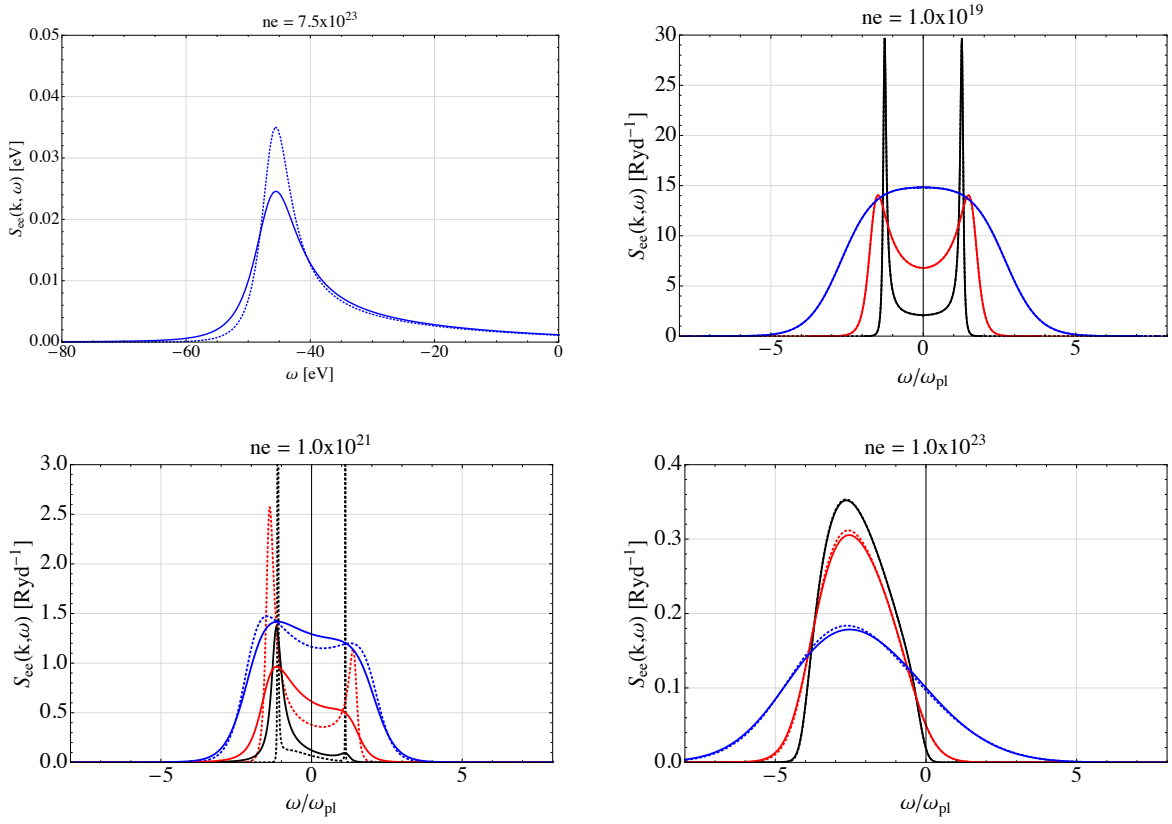


Figure 7: The bound-free feature for beryllium at $k_B T = 20$ eV, density $\rho = 1.85$ g/cm^3, and an initial photon energy of 2960 eV. The figure on the left is for a scattering angle of $\theta = 30°$ and the figure on the right is for $\theta = 150°$. In both cases the feature arises from the 1s bound state. Compare to figure 5 of [3].

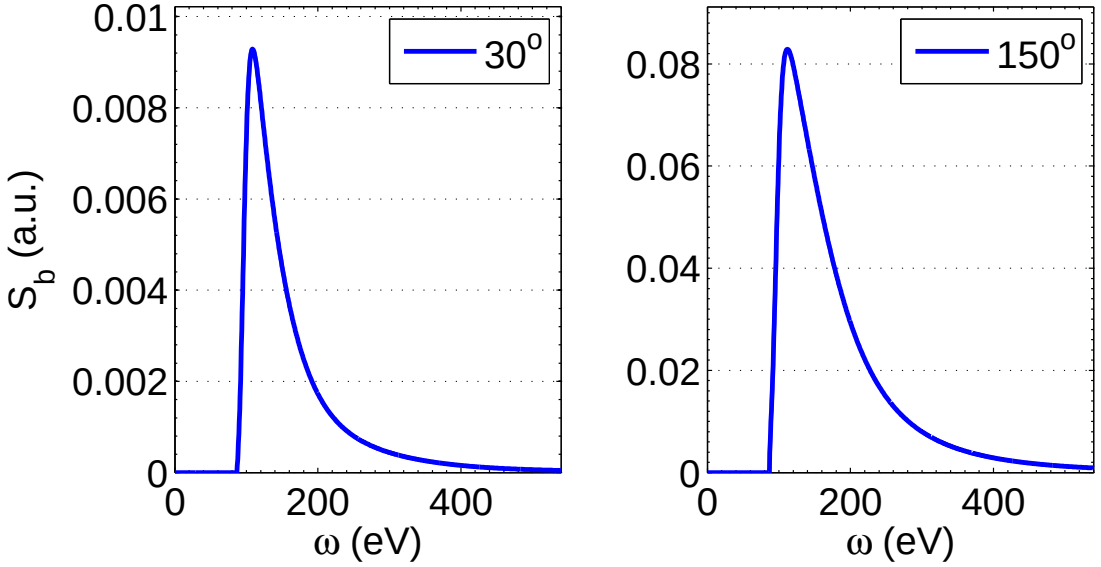


Fig. 7 are slightly smaller. Though it is not completely certain, this may be explained by the inclusion of the factor $1 - \mathcal{F}(\epsilon_p)$ in Eq. (12). This factor is a correction to the model used in [3] and can reduce the low-energy (that is, low-ω) end of the structure function in such a way to lower peaks as seen in this case.

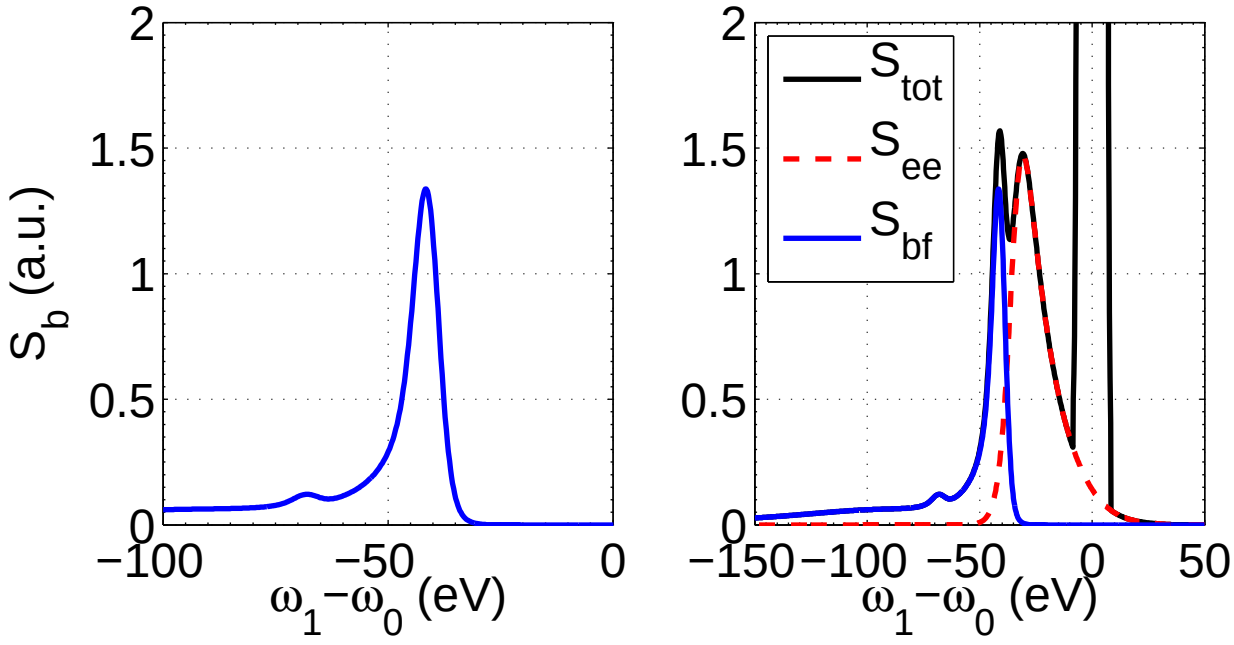
In fact, in the other case considered in this section, a similar discrepancy is even more apparent. Fig. 8 shows the same chromium case shown in §2.4. Note that the case is now plotted against $-\omega = \omega_1 - \omega_0$ for ease of comparison with the same case in Fig. 2 of [4]. The left side shows the bound-free feature S_b alone, while the right side shows S_b, the free-electron feature S_{ee}, and the total structure function S plotted together. The plots are convolved in this case with a normalized Gaussian of full-width at half-maximum 5 eV as described in §3.1; hence the elastic scattering profile is visible as a substantial spike in the structure function at $\omega_1 = \omega_0$. In comparing with Fig. 2 of [4], the bound-free feature in Fig. 8 is around a factor of 2 smaller. It is likely that the $1 - \mathcal{F}(\epsilon_p)$ factor previously mentioned is to blame for this difference as well.

These examples, and in particular the second, highlight a valuable opportunity for validation of the model against experiment. The drastic change in proportion of the bound-free feature between the two models may be useful in identifying strengths or opportunities for improvement of this model.

3.3 Parameter Dependence

In this section the sensitivity of the total structure function to scattering angle and temperature is explored. The unconvolved bound-free and free-electron structure functions are shown here. Since the elastic scattering feature is approximated as a delta function, it can not be effectively plotted together with the bound-free and free-electron features. The values of $G(k)$, the scaling term on the elastic feature delta function as defined in §2.1, are listed in Appendix C for each of the cases that follow, as are various other details of the cases. In all cases the Born-Mermin dielectric function is used to compute the free-electron feature.

Figure 8: The bound-free feature and convolved theoretical spectra for chromium at $k_BT = 5$ eV, density $\rho = 7.19$ g/cm^3, an initial photon energy of 4750 eV, and scattering angle $\theta = 40°$. The figure on the left shows the bound-free feature only and the figure on the right is the convolved theoretical spectra, including all the contributions. The peak height is ~ 525 atomic units. This is the same case as in Fig. 4. Compare to figure 2 of [4].



3.3.1 Angles

In Fig. 9 the free-electron and bound-free features are shown together with their sum. Note that these plots are given for the absolute frequency ω_1 and that the incident photon energy in all cases is $\omega_0 = 2960$ eV. For small angles the free-electron feature is in the collective regime and the bound-free feature is greatly suppressed. As the angles increase the free-electron feature transitions to the noncollective regime while the bound-free feature grows. Between 120° and 140° the transition from collective to noncollective is complete. Note that the coherence parameter α decreases as the angle increases; this is a manifestation of the transition to the noncollective regime.

One feature that seems to be a consequence of the AA+TCP model is the enhancement of the free-electron feature as the scattering angle transitions from 20° to 40°. When compared to the free-electron feature computed with the RPA dielectric function (not shown here) no enhancement was found and the transition to the non-collective regime happened much quicker and was more drastic. The results shown in Fig. 9 did not change as the number of grid points used to calculate the free-electron feature were increased.

The growth of the bound-free feature with angle is typical. The same phenomenon is observed in Fig. 7 and reflects the fact that electrons that are ionized in the scattering process are more likely to scatter photons at large angles than otherwise. The dominant feature in S_b at every angle is the most weakly-bound feature, which is 2p, with ionization energy around 50 eV. Though not easily visible in Fig. 9, the 2s feature (ionization energy ∼80 eV) also appears and contributes to the overall function. The 1s binding energy is too large to allow the feature to be visible in the figure.

3.3.2 Temperatures

Two figures for several temperature ranges are shown plotted against ω_1 for an incident photon energy $\omega_0 = 2960$ eV. In Fig. 10 the XRTS unconvolved theoretical spectrum is shown for a generic example at 30°. As the temperature increases a transition to the collective regime occurs as is characterized by the values of the dimensionless scattering parameter α. Initially, from 5 eV to 10 eV there is an enhancement in the free-electron feature while going from 10 eV to 20 eV and 20 eV to 50 eV there is a suppression. Note that the bound-free feature is multiplied by a scaling factor, hence very small in this regime, as is normally the case for small scattering angles. In all cases the 2p and 2s features contribute as in §3.3.1, though with varying ionization energies. Note that at 50 eV the bound-free feature gains contributions from the extra, very weakly-bound state 3s. This is visible near $\omega_1 = 2960$ eV, where a spike is visible in S_b.

In Fig. 11 the XRTS unconvolved theoretical spectrum is in the noncollective regime for all the temperatures at 130°. As was the case in the 30° case, the spectra becomes "more" non-collective as the temperature is increased; this trend can be seen by examining the value of α. However, unlike the previous case, the free-electron feature decreases monotonically in maximum height. With regards to the bound-free feature, note that it is no longer multiplied by a scaling factor; S_b is much larger in this regime. As was the case before, at $k_BT = 50$ eV the bound-free feature gains contributions from the weakly-bound 3s state.

4 Conclusion

Starrett and Saumon's ab initio model for warm dense matter [1] and Chihara's structure function [2] have been used to calculate the theoretical spectra of an XRTS experiment off of WDM. The work here is the first time an ab initio model has been used to completely determine the theoretical spectra. To do this computation, the elastic, free electron, and bound-free features had to be computed using Chihara's formula, with inputs from Starrett and Saumon's model. This parallels the work done

Figure 9: The unconvolved theoretical XRTS spectra for aluminum at a density of $\rho = 2.7$ g/cm^3 and $k_BT = 10$ eV for an initial photon energy of 2960 eV and several different angles. The dimensionless scattering parameter α is described in §2.2.

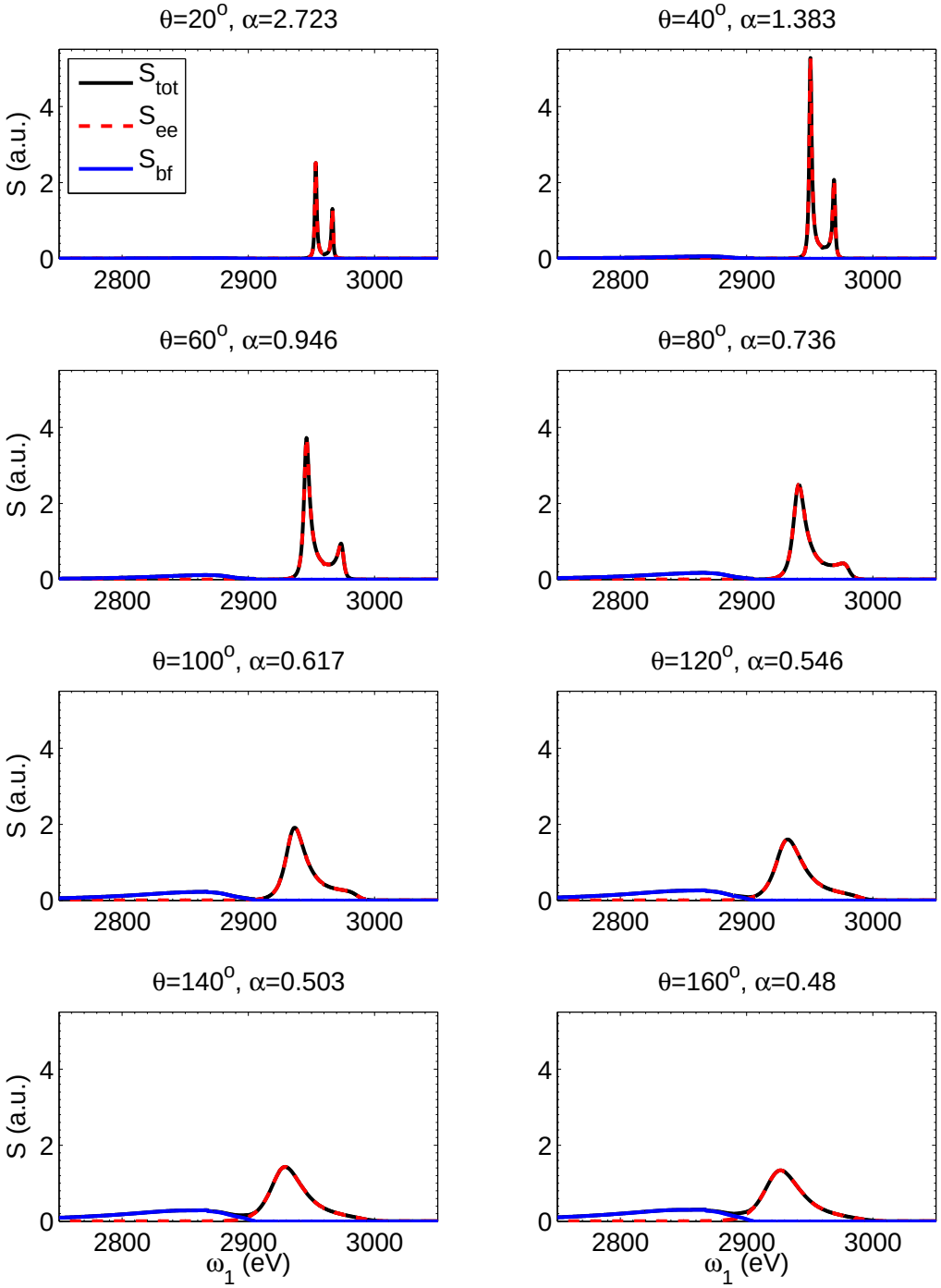
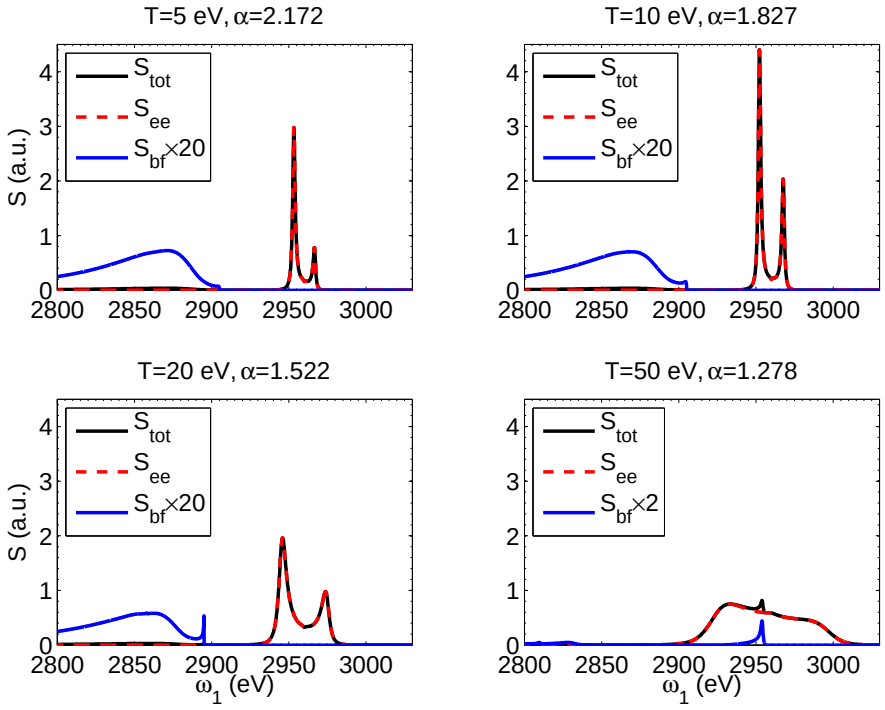


Figure 10: The unconvolved theoretical spectra for aluminum at a density of $\rho = 2.7$ g/cm^3 and several temperatures for an initial photon energy of 2960 eV and scattering angle $\theta = 30°$. The elastic feature is not shown and the dimensionless scattering parameter α is described in §2.2.

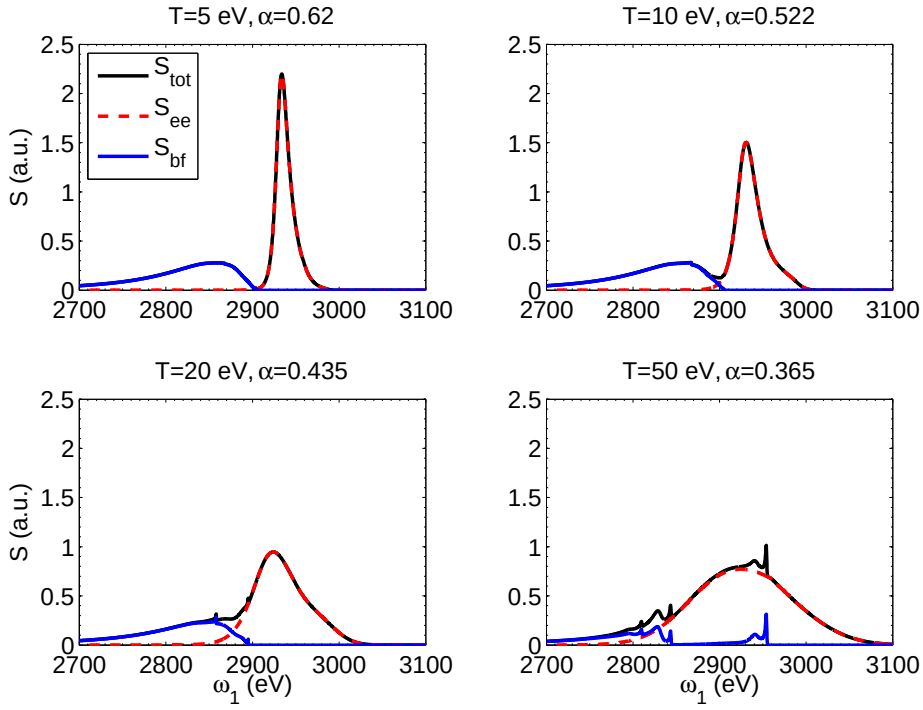


by Johnson et al. [3], but expands it by including a more realistic elastic feature (using Starrett and Saumon's model), goes beyond the RPA approximation for the free-electron feature by using the extended Mermin ansatz and the Born approximation to the dynamic collision frequency, and modifies the bound-free feature to include the occupation of the final state. Furthermore, in order to compare to experiment, it is necessary to convolve the theoretical spectra with an input profile, this has been implemented.

Comparisons have been made to a small part of the existing literature and a preliminary search through parameter space has been performed for different scattering angles and temperatures. In the future, searching through parameter space to find a best fit for given experimental data may provide additional insight as to material properties of WDM.

The next steps of research would involve a thorough comparison with experimental data, further exploration of parameter space to include: sensitivity to initial photon energy and momentum, density of the material, and different materials. Doing such a study will lead to insights as to the validity of the AA+TCP model and Chihara's formula.

Figure 11: The unconvolved theoretical spectra for aluminum at a density of $\rho = 2.7$ g/cm^3 and several temperatures for an initial photon energy of 2960 eV and scattering angle $\theta = 130°$. The elastic feature is not shown and the dimensionless scattering parameter α is described in §2.2.



A Free-electron Feature Details

A.1 RPA Mathematics

Computing the dielectric function can be done analytically to a certain point. Once a simplified expression is obtained, it can then be evaluated numerically. To this end, consider the integral

$$\int_{-1}^{1} \left[\frac{1}{k^2 - 2pk\mu + 2\omega + i\nu} + \frac{1}{k^2 - 2pk\mu - 2\omega - i\nu} \right] d\mu.$$

For notational convenience define the following quantities (with no relation to previous symbols): $\alpha = 2pk$, $\gamma_1 = k^2 + 2\omega$, and $\gamma_2 = k^2 - 2\omega$. With this simplification in place, one can see that

$$\begin{aligned} \alpha \int_{-1}^{1} \left[\frac{1}{\gamma_1 - \alpha\mu + i\nu} + \frac{1}{\gamma_2 - \alpha\mu - i\nu} \right] d\mu &= -\log(\gamma_1 - \alpha\mu + i\nu)\big|_{-1}^{1} - \log(\gamma_2 - \alpha\mu - i\nu)\big|_{-1}^{1} \\ &= \log(\gamma_1 + \alpha + i\nu) - \log(\gamma_1 - \alpha + i\nu) \\ &\quad + \log(\gamma_2 + \alpha - i\nu) - \log(\gamma_2 + \alpha - i\nu). \end{aligned}$$

At this point some care must be taken[3]. Since $\log(z) = \log(x + iy) = \log(\sqrt{x^2 + y^2}) + i \arctan(y, x)$, the real part of logarithm becomes

$$\log\left(\sqrt{\frac{(\gamma_1 + \alpha)^2 + \nu^2}{(\gamma_1 - \alpha)^2 + \nu^2}} \right) + \log\left(\sqrt{\frac{(\gamma_2 + \alpha)^2 + \nu^2}{(\gamma_2 - \alpha)^2 + \nu^2}} \right) \overset{\lim_{\nu \to 0}}{=} \log\left| \frac{k^2 + 2\omega + 2pk}{k^2 + 2\omega - 2pk} \right| + \log\left| \frac{k^2 - 2\omega + 2pk}{k^2 - 2\omega - 2pk} \right|.$$

[3] In what follows the quantity $\arctan(y, x)$ is the two argument arctangent function. This quantity takes into account what quadrant a point is located in before returning the angle. For example, $\arctan(-1, 1) = -\pi/4$ and $\arctan(1, -1) = 3\pi/4$.

case	arctan 1	arctan 2	arctan 3	arctan 4	angle
1	+	+	+	+	$0+0+0+0=0$
2	+	+	+	-	$0+0+0+\pi=\pi$
3	+	+	-	+	$0+0-\pi+0=-\pi$
4	+	+	-	-	$0+0-\pi+\pi=0$
5	+	-	+	+	$0-\pi+0+0=-\pi$
6	+	-	+	-	$0-\pi+0+\pi=0$
7	+	-	-	+	$0-\pi-\pi+0=-2\pi$
8	+	-	-	-	$0-\pi-\pi+\pi=-\pi$
9	-	+	+	+	$\pi+0+0+0=\pi$
10	-	+	+	-	$\pi+0+0+\pi=2\pi$
11	-	+	-	+	$\pi+0-\pi+0=0$
12	-	+	-	-	$\pi+0-\pi+\pi=\pi$
13	-	-	+	+	$\pi-\pi+0+0=0$
14	-	-	+	-	$\pi-\pi+0+\pi=\pi$
15	-	-	-	+	$\pi-\pi-\pi+0=-\pi$
16	-	-	-	-	$\pi-\pi-\pi+\pi=0$

Table 2: All the cases. The + symbol signifies a positive second argument and the − symbol signifies a negative second argument.

(The logarithms are in base e.) Note that this function has four singularities located at the points

$$p=\frac{-k^2-2\omega}{2k}, \qquad p=\frac{k^2+2\omega}{2k},$$
$$p=\frac{-k^2+2\omega}{2k}, \text{ and } \quad p=\frac{k^2+2\omega}{2k}.$$

But since the integration is carried over positive values of p, there are only two singularities that are of concern in the integration, namely,

$$s_1=\left|\frac{2|\omega|-k^2}{2k}\right|,$$
$$s_2=\frac{2|\omega|+k^2}{2k}.$$

Numerically handling the singularities will be the subject of §A.2.

The imaginary part of the logarithm requires careful examination. It is given by the formula

$$\varphi(p;k,\omega,\nu)=\arctan(\nu,\gamma_1+\alpha)-\arctan(\nu,\gamma_1-\alpha)+\arctan(-\nu,\gamma_2+\alpha)-\arctan(-\nu,\gamma_2-\alpha).$$

In the limit that $\nu\to 0^+$ the output of the function is either 0 or $\pm\pi$ or $\pm\pi/2.$, assuming the principal branch of the logarithm is being used. At first glance determining the outcome involves checking $3^4=81$ cases depending on whether or not the second argument in each of the four arctangents are positive, negative, or zero. However, the quantity comes up in an integration over p (which is proportional to α), hence the places where the second argument equals zero are inconsequential. Thus, there are only $2^4=16$ cases to consider depending on the signs of the denominators. In Table 2, all the possibilities are listed for convenience.

In order to determine the value of φ, it is easiest to look for when the second argument changes

signs. This is exactly the same points as the singularities in the real part, namely,

$$s_1 = \left| \frac{2|\omega| - k^2}{2k} \right|$$
$$s_2 = \frac{2|\omega| + k^2}{2k}$$

However, it is necessary to determine which arctangent is undergoes sign flipping. To analyze this situation, the case where $\omega > 0$ will be considered and all the possibilities associated with this case will be exhausted. There are three p ranges to examine, either $p < s_1$, $s_1 < p < s_2$ or $s_2 < p$. Once the various cases are determined for $\omega > 0$, the cases where $\omega < 0$ are discussed.

Suppose $p < s_1$ meaning $\alpha < |\gamma_2|$. Since $\omega > 0$, it is possible to conclude that $\gamma_1 > \gamma_2$, $\gamma_1 > 0$. Now it is necessary to consider possible values for γ_2. First consider the case where γ_2 is positive. This means that $\gamma_2 - \alpha > 0$ and $\gamma_2 + \alpha > 0$ which in turn implies that $\gamma_1 - \alpha > 0$ (remember γ_1 is larger than γ_2). Hence all of the denominators are positive which means that it is case **1** of the table. Hence the value is zero. Now assume that γ_2 is negative. Then $\alpha < -\gamma_2$ which implies that $\alpha + \gamma_2 < 0$. The sign on $\alpha + \gamma_1$ is still positive. Since α and $-\gamma_2$ are positive, it is possible to conclude that $\alpha - \gamma_2 > 0$ meaning $\gamma_2 - \alpha < 0$. For $\gamma_1 - \alpha$, note that $\alpha < |\gamma_2| < \gamma_1$, meaning $\gamma_1 - \alpha$ is also positive. Thus the third denominator is negative, the first is positive, the fourth is negative, and the second is positive. This is case number **4**. In total we can conclude, for $\omega > 0$ and $p < s_1$ the sum of the arctangents is zero.

Now consider the regime where $s_1 < p < s_2$. Since $\omega > 0$ the quantity $s_2 = \gamma_1/2k$ is always positive and hence $\gamma_1 \pm \alpha > 0$. Again there are two cases to consider, depending on the sign of γ_1. Assume that γ_1 is positive, then $\gamma_1 - \alpha < 0$ and $\gamma_1 + \alpha > 0$. The first and second denominator are positive, the third is positive, and the last is negative. This is case **2**. If γ_1 is negative (meaning s_1 is negative) then the first two remain positive while $-\gamma_1 < \alpha$ which means $\alpha + \gamma_1 > 0$ and $\gamma_1 - \alpha < 0$. Hence, regardless of the sign on γ_1, the result is case **2**.

For the last regime $s_2 < \alpha$ observe that the sign in the denominator is the sign of α hence and case **6**.

Thus for $\omega > 0$, in the limit as $\nu \to 0$, the formula for φ, which was originally

$$\varphi(p; k, \omega, \nu) = \arctan(\nu, \gamma_1 + \alpha) - \arctan(\nu, \gamma_1 - \alpha) + \arctan(-\nu, \gamma_2 + \alpha) - \arctan(-\nu, \gamma_2 - \alpha),$$

becomes

$$\varphi(p; k, \omega, 0^+) = \begin{cases} 0 & \text{for } p < |2\omega - k^2|/2k \\ \pi & \text{for } |2\omega - k^2|/2k < p < (2\omega + k^2)/2k \\ 0 & \text{for } (2\omega + k^2)/2k < p \end{cases}.$$

To handle the $\omega < 0$ it is possible to take advantage of the symmetry $\varphi(p; k, \omega, \nu) = -\varphi(p; k, -\omega, \nu)$. This leads to the following formula for φ,

$$\varphi(p; k, \omega, 0^+) = \begin{cases} 0 & \text{for } p < |2|\omega| - k^2|/2k \\ \pi \text{sgn}(\omega) & \text{for } |2|\omega| - k^2|/2k < p < (2|\omega| + k^2)/2k \\ 0 & \text{for } (2|\omega| + k^2)/2k < p \end{cases}.$$

With this simplified formula it is possible to carry out the integration for the imaginary part of the

dielectric function. For illustrative purposes, assume that $\omega > 0$. The integral is given by

$$\begin{aligned}
\mathrm{Im}[\varepsilon(k,\omega)] &= \frac{2}{\pi k^3}\int_0^\infty \mathcal{F}(p^2/;\mu_e,k_BT)p\varphi(p)dp \\
&= \frac{2}{k^3}\int_{s_1}^{s_2} \mathcal{F}(p^2/2;\mu_e,k_BT)pdp \\
&= \frac{2k_BT}{k^3}\log(1+e^{(\mu_e-p^2/2)/k_BT})|_{s_1}^{s_2} \\
&= \frac{2k_BT}{k^3}\ln\left[\frac{1+\exp[(\mu_e-s_2^2/2)/k_BT]}{1+\exp[(\mu_e-s_1^2/2)/k_BT]}\right]
\end{aligned}$$

The total formula (valid for all ω) for the imaginary part of the dielectric function is

$$\mathrm{Im}[\varepsilon(k,\omega)] = \frac{2k_BT}{k^3}\ln\left[\frac{1+\exp[(\mu_e-a(k,\omega)^2/2)/k_BT]}{1+\exp[(\mu_e-b(k,\omega)^2/2)/k_BT]}\right],$$

where

$$\begin{aligned}
a(k,\omega) &= |2\omega-k^2|/2k, \\
b(k,\omega) &= (2\omega+k^2)/2k.
\end{aligned}$$

A.2 RPA Computation

Although it was possible to compute the imaginary part of the dielectric function analytically, this is not the case for the real part. In this section a numerical method to handle the singularities discussed in §A.1 will be shown.

The integrand g in the real part of the dielectric function is a function of p with parameter dependence on k,ω,μ_e, and k_BT is given by

$$g(p;k,\omega,\mu_e,k_BT) = \mathcal{F}(p^2/2;\mu_e,k_BT)p\left[\ln\left|\frac{k^2+2pk+2\omega}{k^2-2pk+2\omega}\right| + \ln\left|\frac{k^2+2pk-2\omega}{k^2-2pk-2\omega}\right|\right].$$

The value of the function g will be abbreviated by $g(p)$. As mentioned previously there are (generally) two singularities in the integrand that may be of concern

$$\begin{aligned}
s_1 &= \left|\frac{2|\omega|-k^2}{2k}\right|, \\
s_2 &= \frac{2|\omega|+k^2}{2k}.
\end{aligned}$$

The first singularity is always less than the second by the triangle inequality. The following integration scheme is used: The integral is divided into 4 regions, the integral from $p=0$ to the first singularity, the first singularity to the second singularity, the second to twice the second, and twice the second to infinity:

$$\int_0^\infty g(p) = \int_0^{s_1} g(p) + \int_{s_1}^{s_2} g(p) + \int_{s_2}^{2s_2} g(p) + \int_{2s_2}^\infty g(p).$$

Numerically, there will be issues if $s_1 \approx 0$ or $s_1 \approx s_2$. This case is handled by setting value of the

integral over $[0, s_1)$ or (s_1, s_2) to zero[4]. Now in each region the following change of variables is used:

$$\begin{aligned}
p &= s_1 \tanh(u) \text{ for } p \in [0, s_1) \\
p &= \frac{(s_2 - s_1)}{2} \tanh(u) + \frac{s_2 + s_1}{2} \text{ for } p \in (s_1, s_2) \\
p &= \frac{s_2}{2} \tanh(u) + 3/2 s_2 \text{ for } p \in (s_2, 2s_2] \\
p &= u \text{ for } p \in [2s_2, \infty)
\end{aligned}$$

The first three mappings effectively place the singularities at infinity (or perhaps one may view it as clustering infinitely many points close to the singularity). This results in the following representation of the integral

$$\begin{aligned}
\int_0^\infty g(p) = &\int_0^\infty g(s_1 \tanh(u))[s_1 \text{sech}^2(u)] du \\
&+ \int_{-\infty}^\infty g\left(\frac{(s_2 - s_1)}{2} \tanh(u) + \frac{s_2 + s_1}{2}\right)\left[\frac{(s_2 - s_1)}{2} \text{ sech}^2(u)\right] du \\
&+ \int_{-\infty}^\infty g\left(\frac{s_2}{2} \tanh(u) + 3/2 s_2\right)\left[\frac{s_2}{2} \text{ sech}^2(u)\right] du \\
&+ \int_{2s_2}^\infty g(u) du
\end{aligned}$$

The first three integrals are handled by using the trapezium rule while the last integral is evaluated using a Gauss-Laguerre integration scheme. Because of the hyperbolic secant factor in the first three integrals, it is only necessary to go to $|u| \approx 15$ before the integrand is extremely small[5]. Furthermore, $\tanh(15) - 1 \approx 10^{-13}$ which is close to machine precision for double precision arithmetic. This means that going any farther than $|u| = 15$ would be effectively the same as evaluating the function at the singularity. However, it is often the case (for numerical stability reasons) that a lower value for $|u|$ must be chosen, for example $|u| \approx 10$ or 12.

A.3 RPA Removable Discontinuity

As mentioned previously, in Eq. 4 there is a removable discontinuity at $\omega = 0$. To see how the cancellations play out, a few Taylor expansions about $\omega = 0$ need to be performed. Observe that

$$\frac{1}{1 - \exp(-\omega/k_B T)} \approx \frac{k_B T}{\omega}.$$

For notational convenience, define two new parameters α and β with $k_B T \alpha = \mu_e - (2\omega - k^2)^2/8k^2$ and $k_B T \beta = \mu_e - (2\omega + k^2)^2/8k^2$ (note that $\alpha \approx \beta$ since $\omega \approx 0$). The following calculation

$$\begin{aligned}
\frac{k^3}{2k_B T} \text{Im}[\varepsilon(k, \omega)] &= \ln\left[\frac{1 + \exp[\alpha]}{1 + \exp[\beta]}\right] \approx -1 + \frac{1 + \exp[\alpha]}{1 + \exp[\beta]} = \frac{\exp[\alpha] - \exp[\beta]}{1 + \exp[\beta]} \\
&= \frac{\exp[\beta](\exp[\alpha - \beta] - 1)}{1 + \exp[\beta]} \approx \frac{\exp[\beta]}{1 + \exp[\beta]}(\alpha - \beta) \\
&= \frac{\exp[\beta]}{1 + \exp[\beta]} \frac{\omega}{k_B T} = \frac{1}{1 + \exp[-\beta]} \frac{\omega}{k_B T} \approx \frac{1}{1 + \exp[-(\mu_e - k^2/8)/k_B T]} \frac{\omega}{k_B T}
\end{aligned}$$

[4] One may also worry about the interval $(s_2, 2s_2]$ but it does not seem to be an issue in practice.

[5] Since the singularities are logarithmic, the rate at which the "g" part grows is much slower than the rate at which the sech decays.

and the observation,

$$\frac{-1}{\varepsilon(k,\omega)} = \frac{\mathrm{Im}[\varepsilon(k,\omega)]}{\mathrm{Re}[\varepsilon(k,\omega)]^2 + \mathrm{Im}[\varepsilon(k,\omega)]^2} \approx \frac{2\omega}{k^3 \mathrm{Re}[\varepsilon(k,0)]^2 + 0} \left(\frac{1}{1+\exp[-(\mu_e - k^2/8)/k_B T]} \right),$$

reveals a simplified formula dielectric component to Eq. 4. Hence the value of Eq. 4 at $\omega = 0$ is

$$S^0_{ee}(k,0) = \frac{k_B T}{\mathrm{Re}[\varepsilon(k,0)]^2} \frac{1}{2\pi^2 n_e k} \left(\frac{1}{1+\exp[(k^2/8 - \mu_e)/k_B T]} \right).$$

For numerical reasons, it is beneficial to set the value of S^0_{ee} to the above quantity once the value of $|\omega|$ falls below some user specified tolerance.

As a check to make sure the dielectric function was being computed correctly, results were compared to calculations in published papers (for example [3] and [5] which are detailed in 3.2), compared to the Lindhard dielectric function, and compared to computations performed in Mathematica. In general good results were obtained with as few as (100, 200, 200, 32) points in the four integration regions, respectively. However, in the code developed to compute the RPA dielectric function (2000, 4000, 4000, 128) points were used.

A.4 Beyond RPA Mathematics

After carefully going through the derivation of the non-complex frequency version, it is straightforward to see how the integral for the real and imaginary part gets modified to include complex frequencies. Denote the real and imaginary part of the Born approximation to the dynamic collision frequency by ν_r and ν_i respectively, in symbols

$$\nu^B = \nu_r + i\nu_i.$$

A lot of the work has already been done in §A.1, if one makes the transcription $\omega \to \omega - \nu_i$ and $\nu \to 2\nu_r$. The justification for this can be seen from the calculation

$$\begin{aligned}
\frac{1}{k^2 - 2pk\mu_e + 2\omega + i\nu} + \frac{1}{k^2 - 2pk\mu_e - 2\omega - i\nu} &\to \frac{1}{k^2 - 2pk\mu_e + 2(\omega + i\nu^B) + i\nu} \\
&\quad + \frac{1}{k^2 - 2pk\mu_e - 2(\omega + i\nu_B) - i\nu} \\
&= \frac{1}{k^2 - 2pk\mu_e + 2(\omega + i\nu_r - \nu_i) + i\nu} \\
&\quad + \frac{1}{k^2 - 2pk\mu_e - 2(\omega + i\nu_r - \nu_i) - i\nu} \\
&= \frac{1}{k^2 - 2pk\mu_e + 2(\omega - \nu_i) + i(2\nu_r + \nu)} \\
&\quad + \frac{1}{k^2 - 2pk\mu_e - 2(\omega - \nu_i) - i(2\nu_r + \nu)}
\end{aligned}$$

and setting the value of ν to zero. Note that the real part of the dynamic collision frequency is always positive.

The arguments from before carry through with the previously mentioned transcription. For conve-

nience define the following quantities

$$g(p;k,\omega,\nu^B) \equiv \log\left(\sqrt{\frac{(k^2+2(\omega-\nu_i)+2pk)^2+(2\nu_r)^2}{(k^2+2(\omega-\nu_i)-2pk)^2+(2\nu_r)^2}}\right) + \log\left(\sqrt{\frac{(k^2-2(\omega-\nu_i)+2pk)^2+(2\nu_r)^2}{(k^2-2(\omega-\nu_i)-2pk)^2+(2\nu_r)^2}}\right)$$

$$\varphi(p;k,\omega,\nu^B) \equiv \arctan(2\nu_r, k^2+2(\omega-\nu_i)+2pk) - \arctan(2\nu_r, k^2+2(\omega-\nu_i)-2pk) + \arctan(-2\nu_r, k^2-2(\omega-\nu_i)+2pk) - \arctan(-2\nu_r, k^2-2(\omega-\nu_i)-2pk)$$

With these definitions the real and imaginary parts of the integral become

$$\mathrm{Re}[\varepsilon^{\mathrm{RPA}}(k,\omega+i\nu^B)] = 1+\frac{2}{\pi k^3}\int_0^\infty \mathcal{F}(p^2/2;\mu_e,k_BT)g(p;k,\omega,\nu^B,\nu)pdp$$

$$\mathrm{Im}[\varepsilon^{\mathrm{RPA}}(k,\omega+i\nu^B)] = \frac{2}{\pi k^3}\int_0^\infty \mathcal{F}(p^2/2;\mu_e,k_BT)\varphi(p;k,\omega,\nu^B,\nu)pdp$$

Although there are no singularities in either integral, if the value of the dynamic collision is small then the real part of the dielectric function will develop sharp cusp-like features and the imaginary part may have finite precision errors. The numerical integration scheme for the above quantities as well as the dynamic collision frequency are discussed in the next section.

A.5 Beyond RPA Computation

The real and imaginary parts of the dielectric function are given by the formulas

$$\nu_r(\omega) = \frac{\Omega_0^2}{24\pi^3 Z_f}\int_0^\infty k^6[V_{ei}^S(k)]^2 S_{ii}(k)\frac{1}{\omega}\left[\mathrm{Re}\left[\varepsilon_e^{\mathrm{RPA}}(k,0)\right]-\mathrm{Re}\left[\varepsilon_e^{\mathrm{RPA}}(k,\omega)\right]\right]dk,$$

$$\nu_i(\omega) = \frac{\Omega_0^2}{24\pi^3 Z_f}\int_0^\infty k^6[V_{ei}^S(k)]^2 S_{ii}(k)\frac{1}{\omega}\mathrm{Im}\ \left[\varepsilon_e^{\mathrm{RPA}}(k,\omega)\right]dk.$$

Both of these are computed using the same integration scheme, a cotangent remapping. This technique transforms an integral of the form

$$\int_0^\infty f(x)dx$$

into

$$\int_0^\pi f(\cot(u/2))\frac{1}{2\sin^2(u/2)}du.$$

The integral above is handled numerically using a trapezoid rule. Note that the value of the function at $u=0$ corresponds to the value of the integrand at infinity, which should be zero. The use of Gauss-Laguerre integration was also considered, but was generally found to be inferior to the cotangent remapping for various cases. Note that computing the dynamic collision frequency involves computing the random phase approximation to the the dielectric function for several values of k, hence it is much more computationally expensive.

To handle the computation of the real and imaginary parts of the random phase approximation to the dielectric function with complex frequencies different schemes were used for the real and imaginary parts. The scheme for the real part resembles the one described in §A.2, except a cotangent remapping is used for the interval from $[2s_2,\infty)$. For the imaginary part a cotangent remapping is used.

As was mentioned previously, computing the dynamic collision frequency can be fairly costly. When computing the RPA dielectric function, the number of points used were (100, 200, 200, 128) in the

respective regions. The amount of time that it takes to compute the dynamic collision frequency was found to scale linearly on the total number of points used to compute the real part of the dielectric function. Also, it scaled linearly with respect to the number of sampled k values in the integrand of the dynamic collision frequency. The RPA dielectric function with complex frequencies was evaluated with no fewer than one thousand integration points for both the real and imaginary parts.

B Computation of the Bound-free Structure Function

Essentially, computing the bound-free structure function amounts to summation over the probabilities of all possible transitions for a given k and ω. Thus, in theory, the double sum in Eq. (12) over values of ℓ_1 and ℓ_2 is infinite. However, the sum can be truncated, because the transition probabilities decrease rapidly as ℓ_1 and ℓ_2 increase. Unfortunately, it is not clear at a glance how many values of ℓ_1 and ℓ_2 are needed to produce a suitably accurate result. Thus the general scheme of computing $S_b(k,\omega)$ is as follows (the infinity norm in step 4 is the single largest magnitude of the pointwise difference between the functions):

1. Set a maximum value $\ell_{\max}$ for ℓ_1 and ℓ_2.
2. Compute Eq. (12) using Eq. (13) and Eq. (10) to obtain S_b^{old}.
3. Increase $\ell_{\max}$ and repeat the calculations in step 2 to obtain S_b^{new}.
4. If $\left\|S_b^{\text{new}} - S_b^{\text{old}}\right\|_\infty$ is sufficiently small, take S_b^{new} as the result. Otherwise, continue to step 5.
5. Set S_b^{new} to be the new S_b^{old}, and repeat steps 3-4.

With a suitable initial choice of $\ell_{\max}$, the process described above generally converges quickly. As can be seen from the scheme, the convergence check currently requires the full function $S_b(k,\omega)$ to be computed for each new value of $\ell_{\max}$. Another scheme, seeking convergence in ℓ_2 for each successive value of ℓ_1, may prove superior, but this has not yet been implemented.

As for the computation of S_b during a single iteration, a few notes are in order. In calculating Eq. (10), the Wigner 3-J symbols are imported from a table generated using the Wolfram Mathematica function `ThreeJSymbol`. The integrand in Eq. (13) is computed as a single function of r before being evaluated via trapezoidal integration with the maximum resolution of points available on the wavefunctions $P_{n\ell}$ and $P_{\epsilon\ell_1}$, which are provided by the AA+TCP model. The computation process is straightforward, but can be rather computationally expensive. This arises from the fact that it consists of a summation over energy states with an integral in r-space at every energy state, at every value of ω to be considered, and at every combination of ℓ_1 and ℓ_2. A few considerations effectively ease the process; first, Wigner 3-J symbols have associated selection rules, which, if not satisfied, render $A_{\ell_1\ell\ell_2}$ zero. Thus, in looping through all possibilities for ℓ_1 and ℓ_2, the integral in r-space and other computations that would go unused are omitted if the selection rules are not satisfied.

Second, the total integrand in Eq. (13) is dominated by the bound state wavefunction beyond a certain threshold r, since the bound state rapidly approaches zero and no other multiplicative terms in the integrand are large. Thus it is possible to identify a point in the bound state wavefunction beyond which the integrand will no longer substantially contribute to the total integral value. This identifies a computational "infinity" up to which to perform the integral. Without choosing such a point, it would be necessary to integrate the integrand on the entire r-array on which the AA+TCP model computes the wavefunctions. Preliminary results show that reducing the integral limit in this way substantially improves speed with no distinguishable consequence to accuracy.

A last optimization focuses not on the individual calculations of $S_b(k,\omega)$ but on judiciously choosing an initial value of $\ell_{\max}$. The time required for an iteration of the bound-free computation is, naturally,

T (eV)	θ (°)	Z_f	μ (a.u.)	α	k (a.u.)	$G(k) = S_{ii}(k)\left\|f(k) + q(k)\right\|^2$ (a.u.)
10	20	2.313	-2.17×10^{-2}	2.723	0.276	28.59
10	40	2.313	-2.17×10^{-2}	1.383	0.543	36.36
10	60	2.313	-2.17×10^{-2}	0.946	0.794	49.56
10	80	2.313	-2.17×10^{-2}	0.736	1.020	64.79
10	100	2.313	-2.17×10^{-2}	0.617	1.216	75.64
10	120	2.313	-2.17×10^{-2}	0.546	1.375	79.61
10	140	2.313	-2.17×10^{-2}	0.503	1.492	79.43
10	160	2.313	-2.17×10^{-2}	0.480	1.563	78.24
5	30	2.143	0.240	2.172	0.411	26.55
10	30	2.313	-2.17×10^{-2}	1.827	0.411	31.76
20	30	2.846	-0.716	1.522	0.411	32.01
50	30	4.730	-3.612	1.278	0.411	22.50
5	130	2.143	0.240	0.620	1.439	84.83
10	130	2.313	-2.17×10^{-2}	0.522	1.439	79.81
20	130	2.846	-0.716	0.435	1.439	72.11
50	130	4.730	-3.612	0.365	1.439	47.99

Table 3: Run parameters and results for the runs discussed in §3.3.

dependent on the current value of $\ell_{\max}$. Greater values of $\ell_{\max}$ imply looping over more values of ℓ_1 and ℓ_2, and thus the computation takes longer. However, smaller choices for an initial $\ell_{\max}$ require more computation iterations before convergence. Thus it is preferable to choose the smallest $\ell_{\max}$ such that the first and second iterations are within the specified tolerance. Unfortunately, there is no firm rule for this choice. A suitable choice for $\ell_{\max}$ depends on the material properties and the specific parameters of the scattering experiment. Materials with a greater number of electrons have more transition possibilities and so larger values of ℓ_1 and ℓ_2 play a more significant role in their bound-free structure function. Hence, as a rough approach to select the correct first value of $\ell_{\max}$, the material atomic number is used. While not perfect, this usually leads to convergence in no more than four iterations. As the number of iterations required seems to depend on k as well, it may be prudent to somehow scale the atomic number by k to improve convergence speed; this remains unimplemented. A complete reformulation of the convergence check, as described earlier in this section, may be preferable.

C Parameter Dependence Case Details

Details on the various runs discussed in §3.3 are shown in Table 3. All runs concern aluminum at solid density ($\rho = 2.7$ g/cm^3) with incident photon energy $\omega_0 = 2960$ eV. The ion number density is the same for all cases: $n_i = 6.026 \times 10^{22}$; the electron density is the product $n_e = Z_f n_i$.

References

[1] C.E. Starrett and D. Saumon, Phys. Rev. E **87**, 013104 (2013)

[2] J. Chihara, J. Phys. Condens. Matter **12**, 231 (2000)

[3] W.R. Johnson, J. Nilsen, and K.T. Chang, Phys. Rev. E **86**, 036410 (2012).

[4] W.R. Johnson, J. Nilsen, and K.T. Chang, High Energy Density Physics **9**, 407, (2013)

[5] S.H. Glenzer and R. Redmer, Phys. Rev. E **81**, 1625 (2009)

[6] B.A. Mattern and G. T. Seidler, Physics of Plasmas 20, 022706 (2013)

Index

The VEX Radiation Module: 2D Radiation Transport with Mimetic Diffusion for ExaFLAG

(Jimmy Fung and Mack Kenamond, mentors)

LA-UR-13-26513
Approved for public release; distribution is unlimited.

Title: THE VEX RADIATION MODULE: 2D RADIATION TRANSPORT WITH MIMETIC DIFFUSION FOR EXAFLAG

Author(s): Powell, Devon M.
Lovegrove, Elizabeth G.
Fung, Jimmy
Kenamond, Mark A.

Intended for: Summer Computational Workshop, 2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)
Report

Issued: 2013-08-16

THE VEX RADIATION MODULE: 2D RADIATION TRANSPORT WITH MIMETIC DIFFUSION FOR EXAFLAG

ELIZABETH LOVEGROVE[1] AND DEVON POWELL[2],
MENTORED BY JIMMY FUNG[3] & MACK KENAMOND[3]

Abstract. The VEX (Variable Eddington for eXaFlag) Radiation Module is a new physics module for the ExaFlag advanced architecture testbed that implements radiation transport coupled to hydro. VEX solves the angle- and frequency-averaged transport equations under the diffusion approximation using a variable Eddington factor, allowing it to more correctly represent behavior in optically thin regimes. It uses a mimetic diffusion solver to solve for the radiation energy density.

1. Introduction

ExaFlag is a testbed for advanced architectures modeled on the FLAG unstructured Lagrangian hydrodynamics code. ExaFlag uses the same unstructured mesh data types as FLAG and has a similar overall design, meaning that innovations developed in ExaFlag can be fed back into FLAG. ExaFlag is written in C++ and is meant to serve as a lightweight code for testing out new ideas in optimization, algorithms, and code design. It can be used to test new algorithms and new potential operator splits without the burden of modifying a large, fully-featured production code like FLAG. For our project during the 2013 Computational Physics Summer Workshop, we implemented a 2D radiation transport module called VEX (Variable Eddington for eXaFlag) that uses a mimetic diffusion solver to advance the radiation energy equation.

2. Equations of Radiation Transport

2.1. **Flux-Limited Diffusion.** We define the radiation energy density as E_r, the radiation flux as the vector $\vec{F}$, and the radiation pressure as the tensor $\mathcal{P}$. The general moments of radiation transport averaged over angle and frequency in the comoving Lagrangian frame to order $\mathcal{O}(v/c)$ are (Buchler 1983, Castor 2004):

$$\frac{\partial E_r}{\partial t} + \nabla \cdot (\vec{u} E_r) = -\nabla \cdot \vec{F} - \mathcal{P} : \nabla \vec{u} + \int (4\pi j_\nu - c k_\nu E_{r,\nu})\, d\nu \tag{1}$$

$$\frac{1}{c}\frac{\partial \vec{F}}{\partial t} + \frac{1}{c}\nabla \cdot (\vec{u}\vec{F}) = -c\nabla \cdot \mathcal{P} - \int k_\nu \vec{F}_\nu \, d\nu \tag{2}$$

These equations are derived by starting from the radiation intensity equation and taking successive moments. Since each moment of the radiation intensity involves the next higher moment, the system can only be closed by assuming a form for the highest moment in terms of the lower ones. The diffusion approximation closes the radiation transport equations by representing radiation flux as a diffusion process with a Fick's Law form:

$$\vec{F} = -D\nabla E_r \tag{3}$$

[1]UC - Santa Cruz Department of Astronomy & Astrophysics
[2]Stanford University Department of Physics
[3]Los Alamos National Laboratory

A closure for $\mathcal{P}$, however, is now necessary. The relationship between E_r and $\mathcal{P}$ is called the Eddington factor f, defined as:

$$f = \frac{\mathcal{P}_{zz}}{E_r} \tag{4}$$

In the simplest diffusion approximation $f = 1/3$, derived by noting that $\mathrm{Tr}[\mathcal{P}] = E_r$ and deep in an optically thick material $\mathcal{P}$ ought to be isotropic. Thus off-diagonal elements of $\mathcal{P}$ should vanish and diagonal elements of $\mathcal{P}$ should be identical. This leads to:

$$\mathcal{P} = \frac{1}{3} E_r \mathcal{I} \tag{5}$$

$$\vec{F} = \frac{1}{3} \frac{c}{\chi_R} \nabla E_r \tag{6}$$

where $\mathcal{I}$ is the identity tensor. We additionally assume that the radiation spectrum has a blackbody form. With these substitutions and simplifications the radiation energy equation in the comoving frame becomes:

$$\rho \frac{D}{Dt} \left[\frac{E_r}{\rho} \right] = \nabla \cdot \left(\frac{c}{3\chi_R} \nabla E_r \right) - \frac{1}{3} \nabla \cdot \vec{u} + c\kappa(aT^4 - E_r) \tag{7}$$

This is the standard Eddington approximation.

The diffusion approximation captures many of the essential features of radiation transport, especially in optically thick regimes, and has the advantage of being simple and computationally tractable. For this reason it is used in many radiation transport codes. The standard Eddington approximation performs well in optically thick regimes and is qualitatively accurate in optically thin. However the dropped terms in the flux equation, as well as the general assumption that the material is optically thick everywhere, will lead to a significant error in the results. This approximation will also lead to the diffusion solver attempting to propagate radiation infinitely fast as $\chi_R \to 0$ and the system becomes optically thin. We can mitigate these errors somewhat while continuing to use the diffusion approximation by varying f as some function of the optical properties of the system to give more correct behavior in the thin limit, hence the name *variable Eddington factor.*

There are many ways of choosing or calculating the form of f. We adopt the method known as *flux-limited diffusion.* We introduce the flux limiter λ in the diffusion coefficient, giving it the form $D = c\lambda/\chi_R$ instead. Then f is calculated as some function of λ. For the form of λ and the corresponding closure of $\mathcal{P}$ we use the analysis of Levermore and Pomraning 1981 and Levermore 1984. These papers define the variable $R = |\nabla E_r|/\chi_R E_r$ and from it compute a spatially-varying λ and f:

$$\lambda = \frac{1}{R} \left(\coth R - \frac{1}{R} \right) \approx \frac{2+R}{6+3R+R^2} \tag{8}$$

$$f = \lambda + \lambda^2 R^2 \tag{9}$$

$$\mathcal{P} = \frac{E_r}{2} \left((1-f)\mathcal{I} + (3f-2)\hat{n}\hat{n} \right) \tag{10}$$

In the optically thick limit, $R \to 0$, $\lambda \to 1/3$ and $f \to \lambda$. The pressure tensor closure then goes to $(E_r/3)\,\mathcal{I}$ and the Eddington approximation is recovered. In the optically thin limit, however, $R \to \infty$, $\lambda \to 0$, $f \to 1$, and the pressure tensor becomes $(E_r/2)\,\hat{n}\hat{n}$. χ_R in the diffusion coefficient $c\lambda/\chi_R$ will go to zero, but λ will also decrease to compensate. The flux is therefore limited to moving at lightspeed rather than attempting to propagate infinitely fast as $\chi_R \to 0$, as it will under the standard Eddington approximation.

Under this approximation, the coupled radiation-hydro equations in the Lagrangian frame become:

$$\rho \frac{D\vec{u}}{Dt} = -\nabla P - \lambda \nabla E_r \tag{11}$$

$$\rho \frac{De}{Dt} = -P\nabla \cdot \vec{u} - c\kappa(aT^4 - E_r) \tag{12}$$

$$\rho \frac{D}{Dt}\left[\frac{E_r}{\rho}\right] = \nabla \cdot \left(\frac{c\lambda}{\chi_R}\nabla E_r\right) - \mathcal{P} : \nabla \vec{u} + c\kappa(aT^4 - E_r) \tag{13}$$

where e is material specific internal energy. These are the equations solved by VEX. Equation 11 is the conservation of momentum equation and now has one force term for the usual pressure gradient and one force term representing momentum exchange with the radiation. Equation 12 is the hydro internal energy equation and contains the usual PdV work term plus a contribution from absorption/emission exchange with the radiation. Equation 13 is the new radiation energy equation, derived in the same way as Equation 7 but without making the $f = 1/3$ substitution.

The notation $\mathcal{P} : \nabla \vec{u}$ indicates a tensor contraction. Expanded out into components this term becomes:

$$\mathcal{P} : \nabla \vec{u} = \mathcal{P}_{ij}\frac{du^j}{dx^i} \tag{14}$$

$$= \mathcal{P}_{xx}\frac{\partial u_x}{\partial x} + \mathcal{P}_{yy}\frac{\partial u_y}{\partial y} + \mathcal{P}_{xy}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \tag{15}$$

since the tensor is symmetric. From the closure of Equation 10 the components of $\mathcal{P}$ become:

$$\mathcal{P}_{xx} = \frac{E_r}{2}\left(1 - f + \frac{(3f-1)}{|\nabla E_r|^2}(\nabla E_{r,x})^2\right) \tag{16}$$

$$\mathcal{P}_{yy} = \frac{E_r}{2}\left(1 - f + \frac{(3f-1)}{|\nabla E_r|^2}(\nabla E_{r,y})^2\right) \tag{17}$$

$$\mathcal{P}_{xy} = \frac{E_r}{2}\left(\frac{(3f-1)}{|\nabla E_r|^2}\nabla E_{r,x}\nabla E_{r,y}\right) \tag{18}$$

When $f \to 1/3$ the cross-term disappears and the diagonal terms become $\mathcal{P}_{xx} = \mathcal{P}_{yy} = E_r/3$, reducing this term to $(1/3)\nabla \cdot \vec{u}$ as in Equation 7.

These equations conserve energy but not momentum, as the source exchange term $c\kappa(aT^4 - E_r)$ is mirrored in both the hydro internal energy equation and radiation energy equation, but the momentum exchange term $\lambda \nabla E_r$ has no counterpart since we do not explicitly track radiation momentum. The use of a variable λ, while improving the accuracy of the method, also introduces additional complications stemming from the fact that the diffusion coefficient is now a function of space and time. Specifically, the momentum coupling term $\lambda \nabla E_r$ can now no longer be folded into the ∇P term as a simple "radiation pressure," since λ can no longer be moved inside the gradient. The tensor contraction $\mathcal{P} : \nabla \vec{u}$ must now be calculated in full instead of being reduced to $(1/3)\nabla \cdot \vec{u}$. However, with the proper operator splits and diffusion solver, these constraints can be handled with minimal complication.

2.2. **Treatment of Terms.**

2.2.1. *Hydrodynamic Coupling Terms.* The equations solved by VEX contain two terms that couple radiation and hydro: one in the momentum equation that represents the transfer of momentum from radiation to gas; and one in the energy equations that represents energy exchange via absorption and emission.

The source term $c\kappa(aT^4 - E_r)$ is quite stiff and, if treated explicitly, will constrain the code to a cooling timescale that is in many cases very small. Therefore we split this term off and treat it

implicitly. Our treatment is based on the work of Morel (unpublished lecture notes) and results in the following equation for material specific internal energy:

$$e^{k+1} = \frac{\frac{\rho_i}{\Delta t} e^k + c\kappa(E_r^k + \frac{3a}{c_V} T^{4,k})}{\frac{\rho_i}{\Delta t} + \frac{4ac\kappa}{c_V} T^{3,k}} \tag{19}$$

where the units of κ are cm^{-1} and a is the radiation constant $4\sigma/c$.

The momentum term $\lambda \nabla E_r$ is not difficult to calculate in itself, but care must be taken when adding it to the hydro momentum equation. If the hydro portion of the code uses a compatible work formulation, as the staggered-grid hydro in ExaFlag does, then if this term is added in the wrong place it will also be included in the hydro energy equation, producing a spurious radiation "work" term. In fact in the comoving frame, if only internal energy is considered, radiation will appear to do no work. Its effects appear instead in the kinetic energy term, and if total energy is instead evolved a radiation term will appear.

2.2.2. *Radiation Energy Density.* The radiation energy density equation is solved as a whole via the mimetic diffusion solver, as discussed in Section 4. The diffusion solver advances an equation of the form:

$$a\frac{\partial u}{\partial t} = \nabla \cdot \mathcal{K} \nabla u + f_D \tag{20}$$

where u is the quantity of interest and f_D is some set of source terms (not to be confused with the Eddington factor f). We therefore calculate both the exchange term $c\kappa(aT^4 - E_r)$ and the radiation pressure term $\mathcal{P} : \nabla \vec{u}$ and combine them into f_D. We then use the current values of E_r to calculate λ and from there the diffusion coefficient $\mathcal{K}$. Then the entire equation is handed off to the diffusion solver along with a timestep, which solves implicitly for the updated value of E_r at the new time.

3. Module Design

VEX's additions to the ExaFlag testbed are contained largely within two new base classes, `TransportBase` and `FieldBase`, and two new classes, `FLDsolver` and `DiffusionFC`.

`TransportBase` is analogous to the hydro class `HydroBase` in that it provides an abstract base class for a physics module, in this case a module capable of solving a transport equation for some quantity. `TransportBase` makes no assumptions about the quantity to be transported (e.g. photons, neutrinos, etc.) nor about the physics implemented to solve the transport equations. `FLDSolver` inherits from `TransportBase` and implements the specific flux-limited diffusion physics and solution method outlined in the previous section. `FieldBase` implements a basic field that tracks radiation state variables such as E_r, but not FLD-specific variables such as λ and f. Thus `FieldBase` is meant to remain agnostic to the transport solver used. `DiffusionFC` implements the mimetic diffusion solver that will be outlined in Section 4. `FLDSolver` keeps an object of this class and makes a call to it to solve the diffusion equation and advance the radiation energy.

`FLDSolver` consists of four main steps interleaved with the staggered-grid hydrodynamics module, `SGHydro`. `SGHydro` currently uses a simple predictor-corrector scheme to advance the hydro variables, but more complicated timestepping schemes such as RK4 could be added with a minimum of effort. In the current form, the FLD solver runs first. First the code calculates R, λ, and f. Then it solves implicitly for the new e^{k+1} updated for the $c\kappa(aT^4 - E_r)$ source term, as shown in Equation 19. We assert that the $\rho\Delta e$ due to this term should exactly equal the ΔE_r due to this term, i.e. any energy lost by the radiation is gained by the material and vice versa. The code stores this Δe in the `FieldBase` object for later use in the hydro. Then it calculates the $\mathcal{P} : \nabla \vec{u}$ term and combines this with the source term to form the diffusion source terms f_D. It computes the spatially-varying diffusion coefficient from λ. Now we have sufficient parameters to perform a diffusion solve. All of these variables are passed to a `DiffusionFC` object that then advances

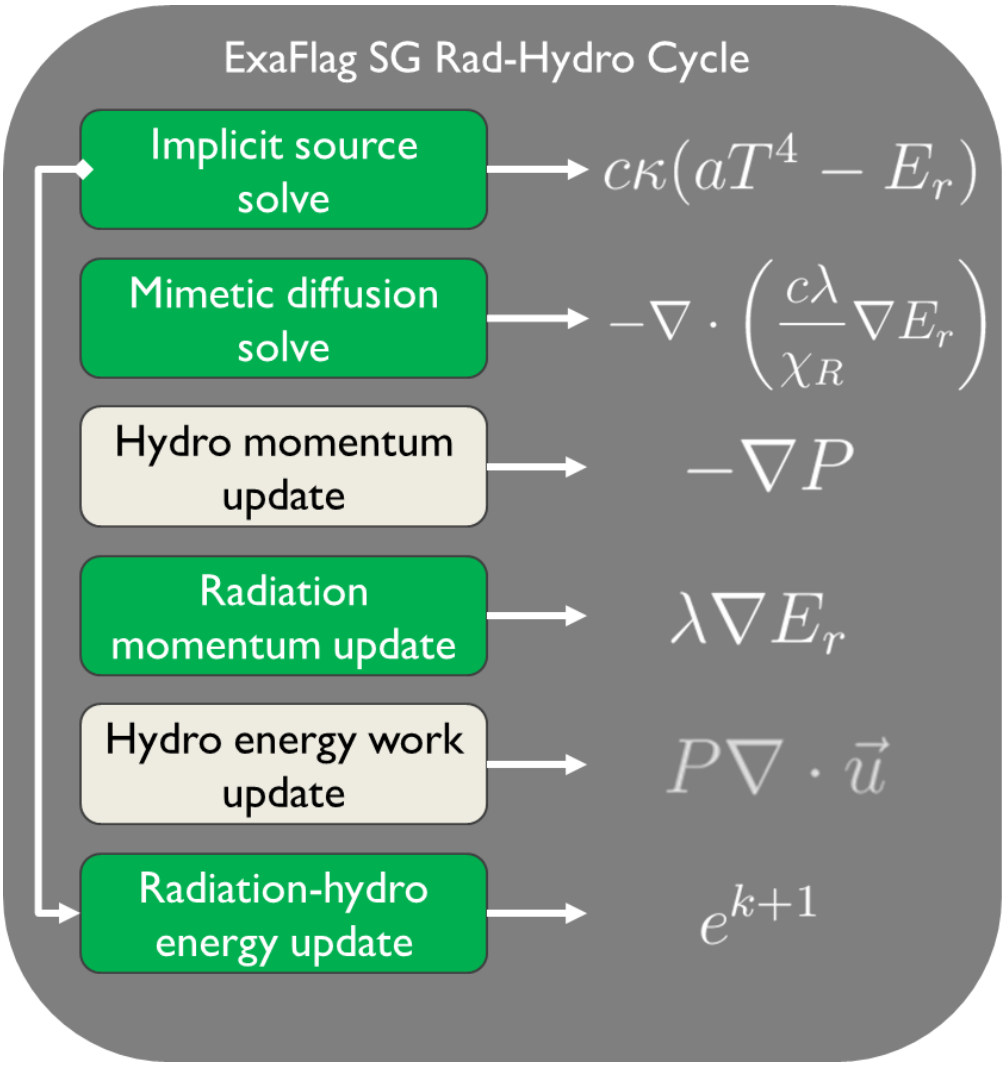


FIGURE 1. Flowchart of one timestep in the ExaFlag rad-hydro staggered-grid cycle.

E_r. `FLDSolver` now recomputes ∇E_r from the updated E_r in order to calculate a new R on the next iteration. Boundary conditions are reapplied to the new ∇E_r. Finally the radiation module computes a Courant-limited timestep recommendation based on a radiation-modified sound speed.

The `SGHydro` module now runs as normal, except that after it has computed force terms but before computing work, the module makes a call to `FLDSolver` requesting the radiation force $\lambda \nabla E_r$. `FLDSolver` adds this term to the point forces used to update the velocity, but *not* to the quark forces used to compute the hydro internal energy, in order to avoid introducing a radiation work term. When `SGHydro` reaches the work update, it adds in the Δe computed from the $c\kappa(aT^4 - E_r)$ source term in `FLDSolver` and stored in the `FieldBase` object before proceeding with the standard work update.

4. MIMETIC DIFFUSION SOLVER

We implemented a mimetic diffusion method for transporting radiation energy in the diffusion approximation of the radiation energy equation (13). In this section, we will discuss the diffusion equation in its most general form:

$$a\frac{\partial u}{\partial t} = \nabla \cdot \mathcal{K}\ \nabla u + f_D \tag{21}$$

where a is the heat capacity (for diffusion of volumetric energy density, $a = 1$), u is the scalar quantity being diffused, t is time, $\mathcal{K}$ is the diffusion coefficient, and f_D is the source term. These quantities can be directly compared to those in Equation 13 to see their relation to the radiation energy terms.

4.1. **Introduction to Mimetic Methods.** Physical equations mathematically embody certain important characteristics of the systems they describe. A canonical example is a conservation law (of energy, mass, momentum, etc.). We want to preserve such characteristics when discretizing a continuum system. Discrete methods that reproduce special properties of the continuum system are said to be "mimetic."

For purposes of presenting the mimetic method for diffusion, it will be useful to recast Equation 21 as

$$a\frac{\partial u}{\partial t} = -\nabla \cdot \mathbf{w} + f_D \tag{22}$$

where $\mathbf{w} \equiv -\mathcal{K}\ \nabla u$ is the flux.

Let us define the operators $\mathbf{G}$ and $\mathbf{D}$ over the spatial domain V bounded by the surface ∂V as

$$\mathbf{D}\mathbf{w} \equiv \begin{cases} \nabla \cdot \mathbf{w} & \text{in } V \\ -\mathbf{w} \cdot \hat{\mathbf{n}} & \text{on } \partial V. \end{cases} \tag{23}$$

$$\mathbf{G}u \equiv -\mathcal{K}\ \nabla u \tag{24}$$

The flux operator $\mathbf{G}$ is then a map from the space of scalars to the space of vectors:

$$\mathbf{G} : s \to \mathbf{v} \tag{25}$$

and the divergence operator $\mathbf{D}$ is a map from the space of vectors to the space of scalars:

$$\mathbf{D} : \mathbf{v} \to s \tag{26}$$

The maps defined by Equations 25– 26 suggest that the operators may have adjoint properties; this is indeed the case:

$$\mathbf{D} = \mathbf{G}^* \tag{27}$$

To prove this, we begin by defining the inner products

$$(a, b)_s \equiv \int_V ab\, dV + \oint_{\partial V} ab\, dS \tag{28}$$

for the space of scalars s and

$$(\mathbf{A}, \mathbf{B})_{\mathbf{v}} \equiv \int_V \mathcal{K}^{-1}\mathbf{A} \cdot \mathbf{B}\, dV \tag{29}$$

for the space of vectors $\mathbf{v}$. We also make note of the divergence theorem:

$$\int_V u\ \nabla \cdot \mathbf{w}\, dV + \int_V \mathbf{w} \cdot \nabla u\, dV = \oint_{\partial V} u\, \mathbf{w} \cdot \hat{\mathbf{n}}\, dS. \tag{30}$$

The action of $\mathbf{D}$ in the inner product is $(\mathbf{D}\mathbf{w}, u)_s$, so with the inner products and divergence theorem, we can show that this is equivalent to the action of $\mathbf{G}^*$:

$$(\mathbf{Dw}, u)_s = \int_V (\mathbf{Dw}) u \, dV + \oint_{\partial V} (\mathbf{Dw}) u \, dS \tag{31}$$

$$= \int_V u \, \nabla \cdot \mathbf{w} \, dV - \oint_{\partial V} u \, \mathbf{w} \cdot \hat{\mathbf{n}} \, dS \tag{32}$$

$$= - \int_V \mathbf{w} \cdot \nabla u \, dV \tag{33}$$

$$= - \int_V \mathbf{w} \mathcal{K}^{-1} \mathcal{K} \nabla u \, dV \tag{34}$$

$$= \int_V \mathbf{w} \cdot \mathcal{K}^{-1} \mathbf{G} u \, dV \tag{35}$$

$$= (\mathbf{w}, \mathbf{G}u)_{\mathbf{v}}, \tag{36}$$

which is the definition of adjointness. This adjoint relationship, $\mathbf{D} = \mathbf{G}^*$, is the property we want to preserve when moving to the discretized version of the diffusion equation.

There are two important points to be made with regard to such a mimetic discretization. First, because it was derived from the divergence theorem, the scalar field u is exactly conserved. Second, because the diffusion operator $\nabla \cdot \mathcal{K} \nabla u$ becomes $\mathbf{DG}u = \mathbf{G}^* \mathbf{G}u$, the resulting matrix equation will be symmetric and positive-definite.

The above was an overview of the motivation for mimetic diffusion methods. For a more rigorous derivation of the operator relationships which includes a treatment of the time derivative $\frac{\partial u}{\partial t}$ and source term f_D from Equation 21, see Shashkov and Steinberg 1996.

4.2. **The `DiffusionFC` Module.** There are many mimetic methods in the literature; the one we implemented is described in Shashkov and Steinberg 1996. It discretizes the flux as a projection onto face normals. This method gives second-order spatial convergence and a symmetric positive-definite (SPD) matrix equation. It can accomodate non-convex and/or tangled meshes while maintaining an SPD matrix; however, the method loses accuracy in these instances. Because the operators are derived from the discrete divergence theorem, the scalar quantity u is exactly conserved. The time-stepping scheme is implicit and first-order. Due to its implicitness, the method properly handles the nonlinear case of the diffusion equation in which the diffusion coefficient $\mathcal{K}$ varies in space and time.

Our implementation in ExaFlag is called `DiffusionFC` (for "face-centered"). `DiffusionFC` can handle Neumann, Dirichlet, and mixed (Robin) boundary conditions contained in the module `DiffBC`. Matrices are stored in compressed sparse row (CSR) format. In addition, `DiffusionFC` has options for various threading and linear solver configurations.

4.3. **Convergence.** We verified spatial convergence rates for the diffusion solver using a test problem defined in Shashkov and Steinberg 1996. The problem is as follows:

$$\frac{1}{v} \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial u}{\partial y} \right) + f_D \tag{37}$$

with

$$v = 300, \; k = \frac{1}{30}, \; f = x^2$$

The boundary conditions are

$$k \frac{\partial u}{\partial y} = 0 \text{ at } y = 0 \text{ and } y = 1, \tag{38}$$

$$u - 2k \frac{\partial u}{\partial x} = 0 \text{ at } x = 0 \text{ and } x = 1 \tag{39}$$

This problem has the steady-state solution

$$u = a + bx + cx^4 \tag{40}$$

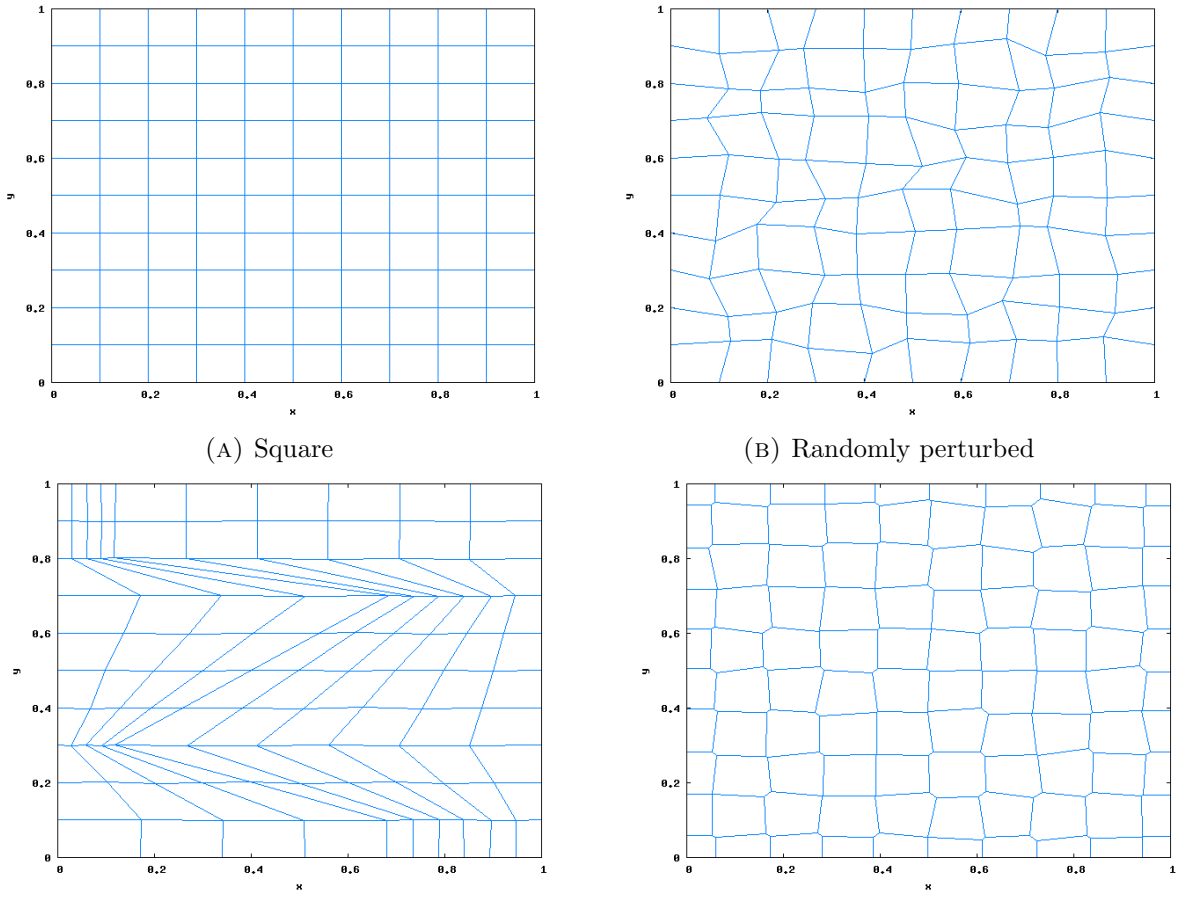
where

$$a = \frac{1}{6}\left(\frac{1+8k}{1+4k}\right),\ b = \frac{1}{12k}\left(\frac{1+8k}{1+4k}\right),\ c = \frac{-1}{12k}$$

Convergence results are shown in Table 1 for the set of test meshes shown in Figure 2. We found that the square, randomly perturbed, and Kershaw meshes perform to second order, as expected for quad meshes. The Voronoi mesh appears not to converge for nonlinear scalar fields, though we did find that for linear fields the Voronoi mesh showed nearly first-order convergence.

In addition, we ran test problems with a linear scalar field (on all quad meshes) and a quadratic scalar field (on the square mesh) to verify exact convergence for these cases. They performed as expected, showing errors consistent with roundoff.

We used a trivial test problem (a constant scalar field with a time-varying source term) to verify first-order temporal convergence. This also worked as expected.



(A) Square

(B) Randomly perturbed

(C) Kershaw

(D) Voronoi

FIGURE 2. Example test meshes ($N = 10$) for the diffusion solver.

4.4. **Unstructured Meshes.**

N	Max. Error	Rate
5	1.17E-1	–
10	3.22E-2	1.86
25	5.44E-3	1.94
100	3.49E-4	1.98
250	5.61E-5	1.99

(A) Square

N	Max. Error	Rate
5	1.38E-1	–
10	3.79E-2	1.87
25	6.91E-3	1.86
100	5.96E-4	1.77
250	9.27E-5	2.03

(B) Randomly perturbed

N	Max. Error	Rate
5	1.73E-1	–
10	5.46E-2	1.66
50	7.69E-3	1.22
150	1.20E-3	1.69
250	4.83E-4	1.79

(C) Kershaw

N	Max. Error	Rate
5	8.01E+0	–
10	9.01E+0	-0.17
25	1.00E+1	-0.11
100	1.10E+1	-0.07
250	1.20E+1	-0.09

(D) Voronoi

TABLE 1. Error data and convergence rates for the diffusion test problem defined in Equations 37–40 on each of the meshes shown in Figure 2.

4.4.1. *Limitations.* There are a few key differences between unstructured and structured meshes. Stencil size and connectivity are unknown *a priori*, so this information must be determined before any element retrieval and matrix assembly occurs. This also affects sparse matrix storage formats like CSR, which require knowledge of the number of nonzero elements per row prior to the insertion of matrix values. Stencil elements are not (in general) contiguous in memory, so memory access patterns are random and inefficient. The matrix assembly process is therefore not easily vectorized. Finally, although the matrix structure is sparse, it will not be banded as matrices yielded by structured mesh methods are. This may have consequences for hardware acceleration and optimization of certain linear solvers.

4.4.2. *Treatment in* `DiffusionFC`. A novel aspect of `DiffusionFC` is that it traverses and stores all of the stencil connectivity data prior to matrix assembly. The original intent was to speed up the code by reducing extraneous pointer-hopping during the matrix assembly process, though after running some tests we determined that there was no appreciable change in execution time. However, we did gain some useful features from a software design perspective.

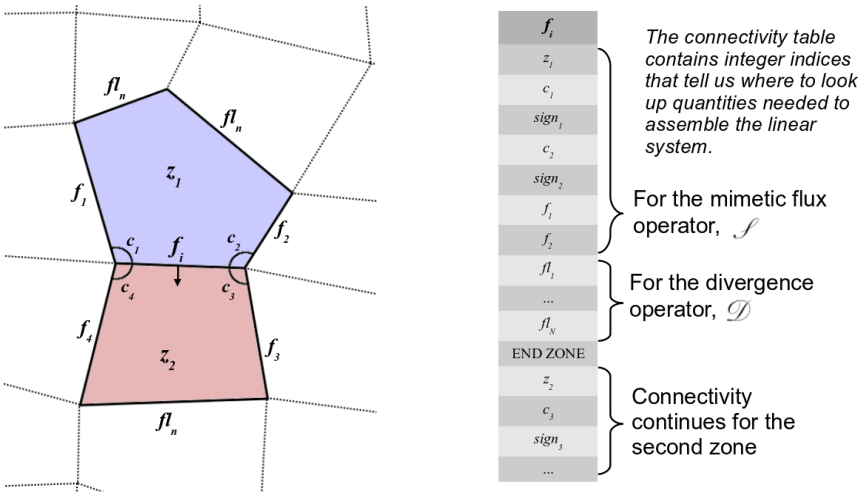
Because this mimetic method is face-centered (see Figure 3a), we need connectivities linking each face to its neighboring corners, zones, and faces. This can only be accomplished by looping over sides, which have explicit connectivity arrays for all three (for a complete description of the mesh structures in ExaFlag and FLAG, see Burton 1992). Face-centered stencil connectivities are thus built using side loops, allowing the rest of `DiffusionFC` to loop over faces in a more natural way. This also made the implementation of threading with OpenMP somewhat easier, as it isolated the threading bottleneck described in Section 4.5.

In addition, by isolating the stencil traversal code (which has multiple nested loops and conditional branches) in a dedicated function, we were able to make the matrix assembly code much cleaner and more readable.

Boundary conditions are now modularized so that the interface is more intuitive. Because this diffusion method requires a global solve, boundary conditions must be applied as a direct modification to many matrix elements. Pre-built connectivity data allow us to cleanly move the application

of boundary conditions into a dedicated container class, `DiffBC`, which can be used in a manner similar to that of a local method like hydrodynamics.

A final benefit of pre-traversing the stencil connectivity is that it we can determine the number of nonzero elements in each row of the matrix, allowing `DiffusionFC` to use CSR-format matrix storage for large problems.



(A) The stencil (B) Connectivity storage

FIGURE 3. (a) The stencil for the mimetic diffusion method and (b) schematic of the resulting connectivity array.

We tried several different storage formats for stencil connectivity, though only two were thread-safe. The first of these was a linked list scheme which proved to be extremely cumbersome and slow to traverse and allocate; this has been removed from the code. The final version uses a hash map in the form of a 2D ragged array, each column of which contains the connectivity data for one face (see Figure 3b). It is fast and clean for traversal, dynamic resizing, and reallocation operations.

4.5. **Acceleration.** Once certain that the `DiffusionFC` was working as designed, we began threading stencil connectivity traversal, coefficient computation, and matrix assembly routines with OpenMP. Without much effort, threading accelerated these routines by a factor of 8–10 with a "sweet spot" around 16–20 threads (see Figure 4).

At this time, there is still an unresolved bottleneck in threading the stencil connectivity traversal. Because this process loops over sides (see Section 4.4.2), the code includes a boolean flag for each face which prevents double-counting. In certain cases, two threads can enter the loop body for a single face before either one has had time to change this flag, causing a crash (this is why were only able to run up to 16 threads on the 25×25 mesh in Figure 4). Using `#pragma omp critical` causes the code to run much slower than even the fully serial case; we are still working on a good way to circumvent this issue.

The linear solve is by far the most expensive part of the diffusion cycle, taking up $> 80\%$ of the total computation time. Various linear solver options are discussed in Section 4.6. This a topic for further investigation.

All of the convergence and timing tests presented here were done with using the Intel PARDISO solver, which despite being a costly direct solve is one of the faster methods we have tried so far. The use of a direct solver for convergence studies allowed us to separate errors in the mimetic method from errors due to the finite convergence tolerance of an iterative solver.

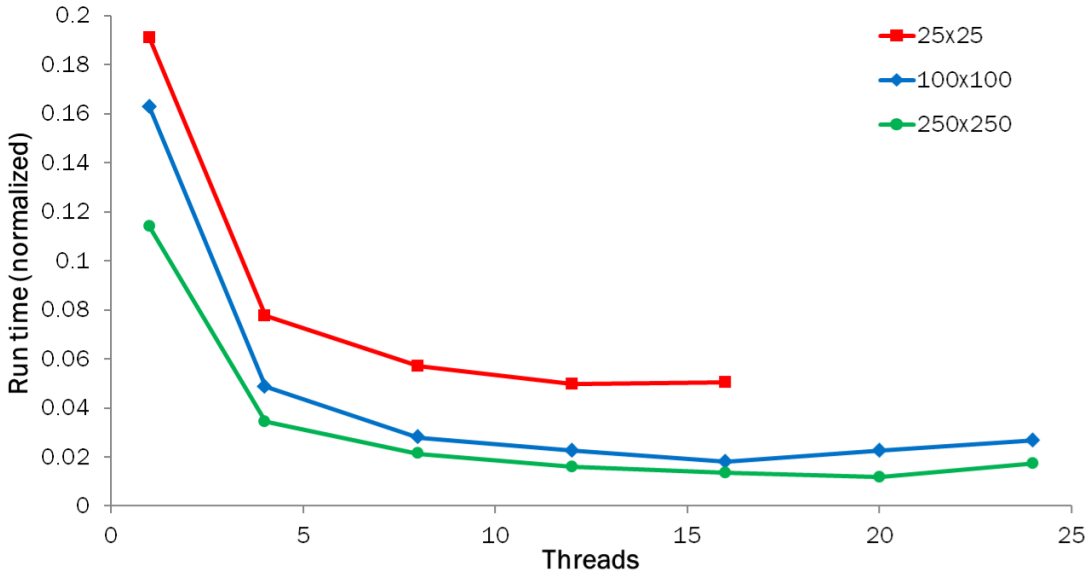


FIGURE 4. Speedup of stencil traversal and matrix assembly with OpenMP. Run times are normalized to the total diffusion cycle time (including the linear solve) for a single thread.

4.6. **Linear solvers.** We implemented several linear solver packages in `DiffusionFC`. They are enumerated here with a short description and comments:

LAPACK `dgesv()`
A direct solver which takes a full, row-major formatted matrix. This was quite easily to implement, but was unsurprisingly slow and impractical for larger systems which necessitate sparse matrix storage. Once better solvers were in place, we removed this option from ExaFlag.

Intel PARDISO
A direct solver which takes a matrix in CSR format. One may set various options regarding the symmetry, definiteness, bandedness, etc. of the matrix. We used the most general (asymmetric indefinite) solver configuration available, though with some implementation changes this could be a promising solver for the symmetric positive-definite matrix given by the mimetic diffusion method. Though it uses an expensive direct method of order $\mathcal{O}(n^3)$, this solver is threaded and quite fast even for large systems.

Hypre BoomerAMG
An algebraic multigrid solver (AMG) from the High Performance Preconditioners library (HYPRE) out of Lawrence Livermore National Laboratory. This solution method is of order $\mathcal{O}(n)$ per iteration, with (ideally) an order of magnitude reduction in residual error at every iteration. As such, BoomerAMG was originally expected to be the best-performing solver in the field, but after some testing it appears that the face-centered solve required by the mimetic diffusion method is unsuitable for multigrid solution, delivering unacceptably slow convergence. In spite of this, the BoomerAMG option is still present in the diffusion solver.

Conjugate gradient
A tried-and-true iterative solver for SPD matrix equations of order $\mathcal{O}(n)$ per iteration. This was implemented directly in the ExaFlag container class `MySPD` using a canned algorithm from Shewchuk 1994 and requires no external modules. It converges quite fast, though there is still much room for improvement. Ideally, this could be extended into a highly parallel, preconditioned method.

Steepest descent
Another iterative solver for SPD matrix equations. This implementation was also taken from Shewchuk 1994. Its convergence is much slower than that of the conjugate gradient solver, so its use is not recommended. Its inclusion in ExaFlag is purely academic.

It is worth noting that iterative solvers can be initialized with the result from the previous diffusion solve. If the problem is slowly varying or close to equilibrium, this drastically reduces the number of iterations needed for the solver to converge.

Chapter 11 of Castor 2004 has a comprehensive overview of linear solve methods for diffusion in a radiation hydrodynamics context.

5. Test Problems for VEX

We feature three main test problems here that demonstrate the source terms, the radiation-hydro coupling, and the variable Eddington factor. All of these tests were conducted on a 100x100 square mesh.

5.1. **Heating & Cooling Problem.** This test problem uses a box of uniform gas and a uniform radiation field that are initialized at different temperatures. The gas velocity is frozen and the only change in its state comes through the source exchange terms with the radiation. In the heating test the gas is started below its equilibrium temperature with the radiation; in the cooling test it is started above it. In both cases the gas temperature should converge to equilibrium with the radiation. The radiation energy density is orders of magnitude larger than the gas energy density and so will remain effectively constant through the test. This problem is solved analytically in Turner and Stone 2001 and serves to test both the accuracy of the source terms and the stability of the implicit solve, as the natural timescale of the cooling test is 10^{-14} s and the code attempts to take 10^{-11} s timesteps.

The parameters used for this test were the same as Zhang et al. 2011: $E_r = 10^{12}$ erg cm^{-3}, $\Delta t = 10^{-11}$ s, and $\rho e = 10^2$ erg cm^{-3} (heating) or $\rho e = 10^{10}$ erg cm^{-3} (cooling). The gas had a density $\rho = 10^{-7}$ g cm^{-3}, a Planck mean opacity $\kappa = 4 \times 10^{-8}$, a mean molecular weight $\mu = 0.6$, and a gamma-law equation of state with $\gamma = 5/3$. It was initialized with zero velocity and not allowed to move for the duration of the test. The results of this test can be seen in Figure 5.

The blue symbols on the figure show the cooling test, the red the heating test. The solid lines denote the analytic solutions. The code correctly heats and cools the gas to its equilibrium temperature using timesteps longer than the cooling timescale. In the case of the cooling test the module significantly undercools for the first few timesteps, but converges to the correct equilibrium value. Iteration of the solve on e and E_r might mitigate this problem and is an area for future work.

5.2. **1D Shocktube Problem.** This test problem demonstrates a shocktube in 1D with radiation coupling. The left side of the domain is initialized at a higher internal energy than the right, causing a shock to form and propagate rightwards. The test problem was again initialized with the parameters of Zhang et al. 2011: uniform gas density of $\rho = 10^{-5}$ g cm^{-3} and uniform zero initial velocity, with the left half of the domain set to $T_L = 1.5 \times 10^6$K and the right half set to

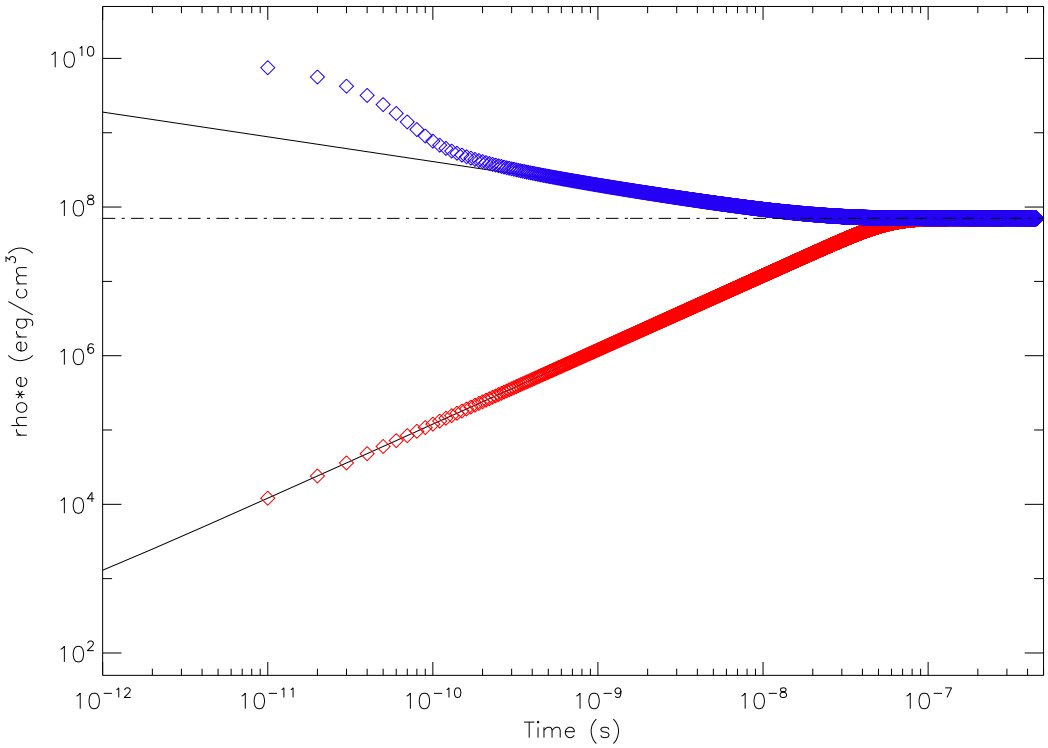


FIGURE 5. Results of gas heating/cooling tests. The red (blue) symbols indicate gas initialized at a lower (higher) temperature than equilibrium. Solid black lines indicate analytic solutions. The dot-dashed line marks the equilibrium gas energy density. In both cases the gas converges on the correct temperature.

$T_R = 3 \times 10^5$ K. The gas is again assumed to have a gamma-law EOS with $\gamma = 5/3$. The mean molecular weight is $\mu = 1$ and the opacities are set to $\kappa = 10^6$ cm^{-1}, $\chi_R = 10^8$ cm^{-1}, thus making the gas and radiation tightly coupled. We therefore expect the gas temperature and radiation temperature to be near equilibrium.

The results are shown in Figure 6. The density spike and rarefaction behind it are clearly discernible in the lower surface. The upper surface shows radiation energy density, which tracks the shock well.

5.3. **Optically Thin Diffusion Problem.** This test problem uses a uniform box of optically thin gas with a radiation pulse initially located at one side of the domain. In the standard Eddington approximation the diffusion coefficient will grow very large and the diffusion solver will propagate this pulse at infinite speed. In the flux-limited diffusion approximation, by contrast, the flux limiter should constrain the pulse to propagate at lightspeed. This test was conducted with the gas held still and radiation forces turned off. Gas density is set to $\rho = 10^{-7}$ g cm^{-3} and opacities are set to $\kappa = 0, \chi_R = 10^{-4}$ cm^{-1}. The timestep is 1.668×10^{-11} s i.e. $\Delta x/(2c)$ and the radiation field is initially at $E_r = 10^{-10}$ erg cm^{-3}, excepting the region adjacent to the leftmost boundary which is set to $E_r = 1$ erg cm^{-3}.

The results can be seen in Figure 7. The golden line marks the correct location of a pulse traveling at lightspeed at the indicated time. The radiation pulse is moving at approximately the correct speed; by contrast, the same test conducted with $\lambda = 1/3$ results in a completely flat radiation field

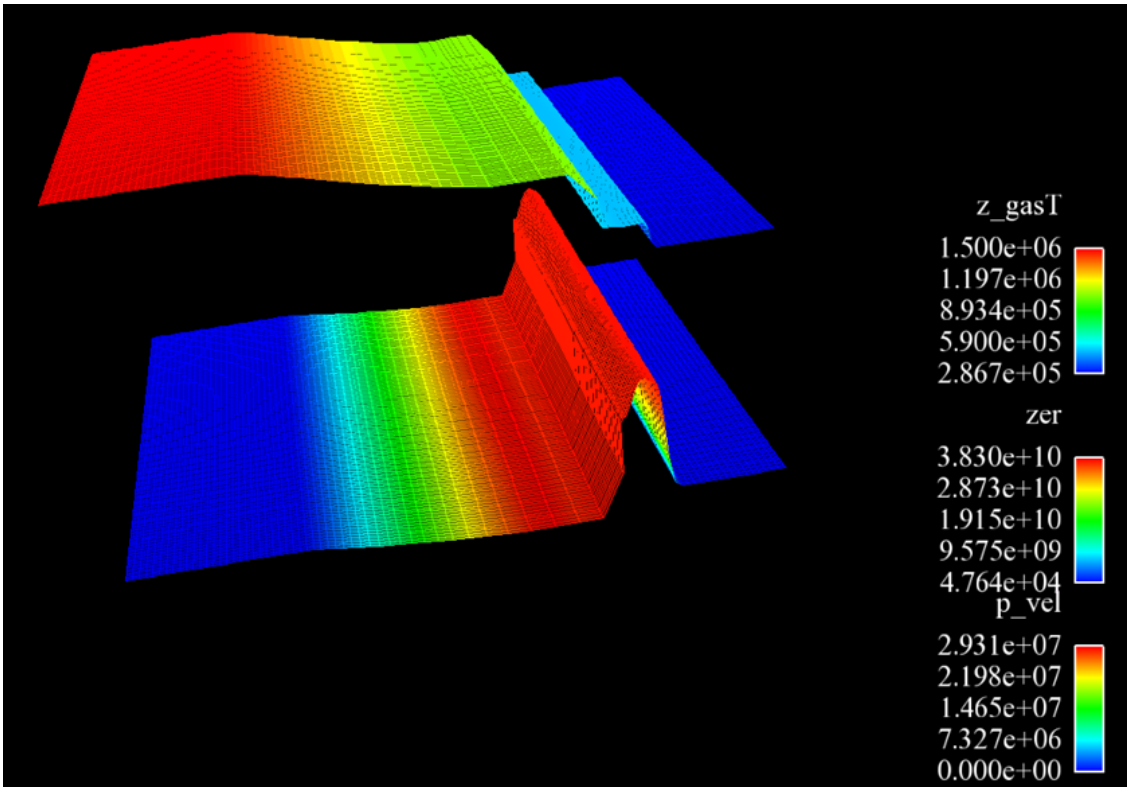


FIGURE 6. Ensight visualization showing a 1D shocktube problem. The lower surface height indicates density, the lower surface color indicates magnitude of velocity. The upper surface height and color indicates radiation energy density. The shocktube shows the expected density spike and rarefaction. The radiation energy tracks the shock.

after one timestep. Thus we have demonstrated that the variable flux limiter is correctly limiting radiation to propagating at lightspeed.

6. Future Work

6.1. **Cell-centered Hydro.** Currently VEX is implemented for the staggered-grid hydro module in ExaFlag. We would like to extend it to also work with the existing cell-centered hydro module. However, the cell-centered method raises interesting concerns regarding the correct operator splitting with radiation. Should the radiation momentum term be included in the Riemann-like vertex solve for velocity, or can this term be added later? Staggered-grid hydro tracks only internal energy, but cell-centered hydro tracks internal, kinetic, and total energy; as a result care must be taken to ensure that all radiation energy deposition is accounted for. Further work is needed to outline the correct operator splitting procedure.

6.2. **Physics Improvements.** A few improvements are possible to the physics of the code to make it applicable to a wider range of problems. The most immediately useful would be allowing temperature-dependent opacities and more complex equations of state. Currently the code relies on fixed opacities and a gamma-law equation of state. As the opacity is already treated as a spatial array within the code, temperature-dependent opacities could be implemented with a minimum of effort. For more complex work, however, both the opacity and EOS routines should eventually be

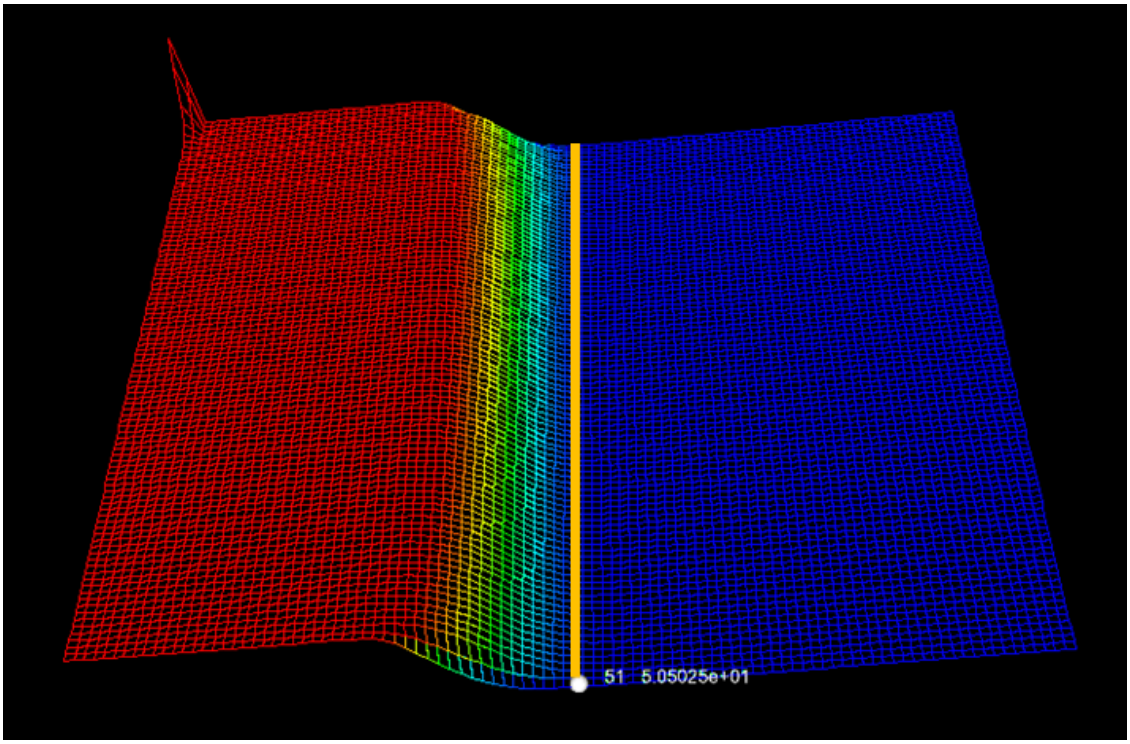


FIGURE 7. Ensight visualization of lightspeed diffusion test. The height and color of the surface indicate radiation energy density. The gold line marks the correct propagation distance for a pulse traveling at lightspeed.

replaced by calls to packages capable of handling more complex physics such as spectral lines and radiation-modified sound speeds.

6.3. **Multiple Fields.** `FLDSolver` currently handles only one field, but it has been designed to make it easy to upgrade to more. The major change involved would be expanding the class to store multiple R, λ, and f arrays. This capability could let the VEX module handle other transported quantities such as neutrinos, or possibly form a path towards a multigroup version of the code. In the multigroup case, however, the class would need to be extended with the ability to couple energy exchange between fields. Multigroup diffusion through clever coupling of source terms is also a possibility.

6.4. **Improvements to FLD & Further Variable Eddington Methods.** The form of λ used in `FLDSolver` is that of Levermore and Pomraning 1981, but other forms are possible, such as the Minerbo solution. These alternate equations have the same limiting behavior but different thick-thin transitions, motivated by different physics. Though these variations tend to have only a small effect on the physical behavior of a problem, the right choice of λ can affect the numerical stability of a simulation and it would be possible to implement the choice of flux-limiter form as an input parameter. Flux-limited diffusion is also not the only variable Eddington method. More complex physics can be implemented by, for example, solving a full transport equation along certain directions with fixed temperature in order to probe the variation in f, then feeding this information back into the diffusion solve.

6.5. **Optimization and Parallelization.** Domain decomposition is a clear next step in having ExaFlag live up to its name. As far as `DiffusionFC` is concerned, this would speed up stencil traversal and matrix assembly as well as facilitating the use of a highly parallel linear solver.

Finding a way to vectorize operations on mesh elements not contiguous in memory (by playing clever tricks with the compiler, for instance) could also further accelerate the stencil traversal and matrix assembly steps of the diffusion solve.

6.6. **Linear Solver.** As HYPRE appears not to work well with the face-centered mimetic method, it seems that a parallel preconditioned conjugate gradient solver is the best bet for speeding up the linear solve process.

6.7. **Mesh Refinement.** As of now, ExaFlag imports mesh data from text files generated in FLAG. This does not easily allow for adaptive mesh refinement. This could be implemented by modifying the connectivity structures in `DelfiMesh`, possibly with the use of a compact hash algorithm for rebuilding connectivity.

6.8. **Improvements to the Mimetic Method.** The mimetic method used in `DiffusionFC` is a basic one designed for quad meshes, which is a good starting point for the implementation of more advanced features. Though `DiffusionFC` does not support tensorial diffusion coefficients ($\mathcal{K}$ is assumed to be of the form $k\mathcal{I}$), it would not be difficult to implement them as described in Hyman et al. 1997. Even more recent methods give good convergence for arbitrary polyhedral meshes (Brezzi et al. 2005), and even allow for meshes with curved faces (Brezzi et al. 2007).

`DiffusionFC` has room for additional bells and whistles unrelated to the method itself. For example, additional options could allow for higher order predictor-corrector timestepping. This would require multiple matrix solves per cycle as well as storage for intermediate solutions.

7. Conclusions

We have implemented a 2D grey flux-limited diffusion radiation module for the ExaFlag testbed that uses a mimetic diffusion solver to advance the radiation energy, verifying the qualitatively correct behavior of this code in the diffusion approximation.

8. Acknowledgements

We acknowledge Misha Shaskov for help with the diffusion solver, Markus Berndt for help with Hypre, Matt Bement for providing test meshes, and Scott Runnels for organizing the workshop. We thank Rob Lowrie, Nathaniel Morgan and Scott Runnels for useful discussions on cell-centered hydro. We also thank Hot Rocks Cafe and the Morning Glory Baking Co. for valuable support in the form of coffee and breakfast burritos. Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396

References

Franco Brezzi, Konstantin Lipnikov, and Valeria Simoncini. A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Methods and Methods in Applied Sciences*, 15(10):1533–1551, 2005.

Franco Brezzi, Konstantin Lipnikov, Mikhail Shashkov, and Valeria Simoncini. A new discretization methodology for diffusion problems on generalized polyhedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 196(3740):3682 – 3692, 2007. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2006.10.028. URL `http://www.sciencedirect.com/science/article/pii/S0045782507000965`. Special Issue Honoring the 80th Birthday of Professor Ivo Babu.

J. R. Buchler. Radiation transfer in the fluid frame. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 30:395–407, November 1983. doi: 10.1016/0022-4073(83)90102-4.

D. E. Burton. Connectivity structures and differencing techniques for staggered-grid free-Lagrange hydrodynamics. In H. J. Shih, W. Schiesser, and J. Ellison, editors, *Presented at the 7th International Association of Mathematics and Computer Simulation (IMACS) International Conference on Computer Methods for Partial Differential Equations, New Brunswick, NJ, 22-24 Jun. 1992*, volume UCRL-JC-110555, pages 22–24, June 1992.

J.I. Castor. *Radiation Hydrodynamics*. Cambridge University Press, 2004. ISBN 9780521540629. URL `http://books.google.com/books?id=J48PULzdgSgC`.

James Hyman, Mikhail Shashkov, and Stanly Steinberg. The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials. *Journal of Computational Physics*, 132(1):130–148, March 1997. ISSN 0021-9991. doi: 10.1006/jcph.1996.5633. URL `http://dx.doi.org/10.1006/jcph.1996.5633`.

C. D. Levermore. Relating Eddington factors to flux limiters. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 31:149–160, February 1984. doi: 10.1016/0022-4073(84)90112-2.

C. D. Levermore and G. C. Pomraning. A flux-limited diffusion theory. *Astrophysical Journal*, 248: 321–334, August 1981. doi: 10.1086/159157.

James Morel. Lagrangian solution of the radiation-hydrodynamics equations with grey radiation diffusion. Unpublished lecture notes.

Mikhail Shashkov and Stanly Steinberg. Solving diffusion equations with rough coefficients in rough grids. *Journal of Computational Physics*, 129(2):383 – 405, 1996. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1996.0257. URL `http://www.sciencedirect.com/science/article/pii/S0021999196902570`.

Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

N. J. Turner and J. M. Stone. A Module for Radiation Hydrodynamic Calculations with ZEUS-2D Using Flux-limited Diffusion. *The Astrophysical Journal Supplement Series*, 135:95–107, July 2001. doi: 10.1086/321779.

W. Zhang, L. Howell, A. Almgren, A. Burrows, and J. Bell. CASTRO: A New Compressible Astrophysical Solver. II. Gray Radiation Hydrodynamics. *The Astrophysical Journal Supplement Series*, 196:20, October 2011. doi: 10.1088/0067-0049/196/2/20.

Verification Problems for xRAGE Radiative Hydrodynamics Code

(Scott Ramsey, mentor)

Verification Problems for xRAGE Radiative Hydrodynamics Code

Elizabeth Hanson, Joseph Redford

Abstract

A new analytical solution was derived for the radiative hydrodynamics equations of a gamma law ideal gas given in [1]. These solutions were derived assuming the velocity distribution is of the form

$$u(r,t) = \frac{\eta r}{t} \tag{1}$$

where η is a constant. In addition, the radiation flux was assumed to be non-divergent.

The analytic solution along with several cases of Coggeshall solution 9 from [1] were then implemented as test cases in the xRAGE radiative hydrodynamics code developed at Los Alamos National Laboratory. By comparing the results of the xRAGE numerical solver with the discretized versions of the analytical solution, we were able to verify the accuracy of specific xRAGE modules.

1 Introduction

Numerical analysis is useful in physics since most real world problems are too complex to admit analytical solutions. This is often the case is in fluid dynamics, and one code for handling numerical radiative hydrodynamics problems is xRAGE. xRAGE is a radiative hydrodynamics solver that was developed at Los Alamos National Laboratory (LANL) and is used for a variety of simulation purposes including modelling stellar interiors and asteroid destruction [2]. In order to ensure that xRAGE produces physically meaningful results, it must be subjected to extensive code verification tests. A simple way of doing this is to model systems with analytical solutions in xRAGE and then compare the output to the analytical solution.

We sought analytical solutions to a set of radiative hydrodynamics equations for an ideal gas published by S. Coggeshall along with 22 analytical solutions in [1]. We used the ninth of the published solutions along with a new one in our verification of xRAGE.

2 Theory and Background

From the general equations of radiative hydrodynamics, as found in [3], S. Coggeshall([1]) used the assumptions of

$$\begin{aligned} P &\approx P_{fluid} \\ p_{rad} &\approx 0 \\ E &\approx E_{fluid} \end{aligned}$$

where P is the pressure, p is the momentum, and E is the internal energy density. The subscript $fluid$ signifies that the quantity is specific to the fluid and rad means it is specific to the radiation. The solutions are assumed to exhibit symmetry such that only one dimension need be considered: [1]

solved the equations for Cartesian, cylindrical, and spherical geometries. Using these assumptions, equations (2), (3), and (4) were obtained. For details of the derivation, please refer to Appendix A.

$$0 = \rho_t + u\rho_r + \rho u_r + \frac{k\rho u}{r} \tag{2}$$

$$0 = u_t + uu_r + \frac{\rho_r}{\rho}\Gamma T + \Gamma T_r \tag{3}$$

$$0 = \frac{\Gamma}{\gamma - 1}(T_t + uT_r) + \Gamma T u_r + \Gamma T \frac{ku}{r} + \frac{1}{\rho}(F_r + \frac{kF}{r}) \tag{4}$$

where ρ is the density, u is the velocity, T is the temperature, F is the heat flux, and k is a geometrical constant that is 0 in Cartesian, 1 in cylindrical, and 2 in spherical coordinates. Along with this, the heat flux was approximated using Fick's Law of diffusion and blackbody radiation, which produces

$$F = -\frac{c\lambda}{3}\nabla a T^4 \tag{5}$$

where c is the speed of light, a is the radiation constant of blackbody radiation, and λ is the mean free path of a photon. The mean free path can be approximated as

$$\lambda(\rho, T) = \lambda_0 \rho^\alpha T^\beta \tag{6}$$

where λ_0, α, and β are constant parameters based on the properties of the fluid. Thus, the overall heat flux in the 1D geometry is

$$F = -\left(\frac{4}{3} c\lambda_0 \rho^\alpha a T^{3+\beta}\right) \nabla T \tag{7}$$

as can be found in [1].

Recognizing that 10 of the 22 solutions in [1] have the form $u \propto \frac{r}{t}$ (solutions 1, 2, 8, 9, 11, 13, 15, 17, 21, and 22), we began by considering solutions with that velocity distribution.

$$\begin{aligned} u &= \frac{\eta r}{t} \\ 0 &= \rho_t + \frac{\eta r}{t}\rho_r + \rho\frac{\eta}{t} + \frac{k\rho\eta}{t} \\ &= \rho_t + \frac{\eta r}{t}\rho_r + \frac{\eta(k+1)}{t}\rho \end{aligned} \tag{8}$$

Now applying a change of variables $\rho(r(\tau), t(\tau))$, we observe that

$$\frac{\partial \rho}{\partial \tau} = \frac{\partial \rho}{\partial t}\frac{\partial t}{\partial \tau} + \frac{\partial \rho}{\partial r}\frac{\partial r}{\partial \tau} \tag{9}$$

We choose $\frac{\partial t}{\partial \tau} = 1$ and $\frac{\partial r}{\partial \tau} = \frac{\eta r}{t}$, which leads to the following:

$$\begin{aligned} \frac{\partial t}{\partial \tau} &= 1 \\ t &= \tau + c \end{aligned}$$

$$
\begin{aligned}
\frac{\partial r}{\partial \tau} &= \frac{\eta r}{t} \\
&= \frac{\eta r}{\tau + c} \\
\frac{\partial r}{r} &= \eta \frac{\partial \tau}{\tau + c} \\
ln(r) &= \eta ln(\tau + c) - ln(d) \\
r &= d^{-1}(\tau + c)^{\eta} \\
\tau + c &= (dr)^{\frac{1}{\eta}} \\
0 &= \rho_t + \frac{\eta r}{t}\rho_r + \frac{\eta(k+1)}{t}\rho \\
&= \rho_\tau + \frac{\eta(k+1)}{t}\rho \\
&= \rho_\tau + \frac{\eta(k+1)}{\tau + c}\rho \\
-\rho_\tau &= \frac{\eta(k+1)}{\tau + c}\rho \\
-\frac{\partial \rho}{\rho} &= \frac{\eta(k+1)\partial\tau}{\tau + c} \\
-ln(\rho) &= \eta(k+1)ln(\tau + c) + e \\
\rho &= e(\tau + c)^{-\eta(k+1)}
\end{aligned}
$$

Here c, d, and e are arbitrary constants of integration. Since $\tau + c$ is equal to both t and $(dr)^{\frac{1}{\eta}}$, and the equation is linear, the general solution is

$$\rho(r,t) = \sum_{i=1}^{n} A_i \Big(\sum_{j=1}^{p} \sum_{k=1}^{q} \sum_{m=1}^{s} \left(x_{ij}(l_{ijk}r)^{\frac{1}{\eta}} + (1 - x_{ij})t \right) f_{ijkm} \left(\frac{t}{(l_{ijk}r)^{\frac{1}{\eta}}} \right) \Big)^{-\eta(k+1)} \tag{10}$$

where f_{ijkm} is an arbitrary function, $n, p, q, s \geq 1,\ n, p, q, s \in \mathbb{N}$, and $x_{ij} \in \mathbb{R}$. It is from here that our solutions started to differ.

2.1 Solution with $\alpha = 0, \beta = -3$

With $\alpha = 0, \beta = -3$ the radiation diffusion becomes

$$F = -\frac{4}{3} c\lambda_0 a \nabla T = -DT_r \tag{11}$$

where D is a diffusion constant. We looked for solutions with $\nabla \cdot F = 0$, and found

$$
\begin{aligned}
F_r + \frac{kF}{r} &= 0 \\
\frac{\frac{\partial}{\partial r}(Fr^k)}{r^k} &= 0 \\
\frac{\partial}{\partial r}(Fr^k) &= 0 \\
Fr^k &= f(t) \\
F &= \frac{f(t)}{r^k}
\end{aligned}
$$

But $\nabla \cdot (r^{-k}\hat{r})$ is $2k\pi\delta(r)$ and not 0 for $k \neq 0$ (which we were primarily interested in), so

$$\begin{aligned} F &= 0 \\ -DT_r &= 0 \\ T &= f(t) \end{aligned}$$

where f is an arbitrary function. The case of $F = \frac{f(t)}{r^k}$ is considered in Appendix B.

Next we consider a simplified form of the density solution $\rho = \frac{j(\frac{r}{t^\eta})}{t^{\eta(k+1)}}$ where again j is an arbitrary function. Substituting these forms for u, ρ, and T in (3), ρ and T were found to be

$$\begin{aligned} \rho &= \frac{A}{t^{\eta(k+1)}} Exp[-\frac{(\eta^2-\eta)r^2}{2\Gamma B t^{2\eta}}] \\ T &= Bt^{2\eta-2} \end{aligned}$$

By using these in equation (4), a restriction on η was found to be

$$\eta = \frac{2}{2+(k+1)(\gamma-1)} \tag{12}$$

So the new solution is

$$\rho = \frac{A}{t^{\eta(k+1)}} Exp[-\frac{(\eta^2-\eta)r^2}{2\Gamma B t^{2\eta}}] \tag{13}$$

$$T = Bt^{2\eta-2} \tag{14}$$

$$u = \frac{\eta r}{t} \tag{15}$$

$$\eta = \frac{2}{2+(k+1)(\gamma-1)} \tag{16}$$

which we call the linear velocity, isothermal solution. Throughout the rest of the paper this solution shall be referred to as the isothermal solution for convenience.

2.2 Coggeshall Solution 9

The other solution we derived using our initial assumptions about the velocity distribution and non-divergent flux is identical to one of the solutions published in Coggeshall's paper. As described in the previous section, linear combinations of $\rho(r,t)$ with varying real parameters A, x, l, and η should be assumed. In the developments to follow, however, we leave off the summations for simplicity and take a single case. We begin with

$$\rho(r,t) = \frac{At^{\eta\chi(k+1)}}{(x(lr)^{\frac{1}{\eta}} + (1-x)t)^{\eta(k+1)}(lr)^{\chi(k+1)}} \tag{17}$$

The factor χ is introduced in the exponent to allow an additional level of flexibility. At this point it is valuable to recall that we are looking for solutions to the flux equation such that $F = \frac{f(t)}{r^k} = -\frac{4ac\lambda_0}{3}\rho^\alpha T^{3+\beta}T_r$. We notice that this expression for flux involves separable variables, and one way for this to be satisfied is by requiring that both $\rho(r,t)$ and $T(r,t)$ be separable.

In order to obtain a separable equation for the density, we must have $x = 0$ or $x = 1$, which results in the following expressions:

$$\begin{aligned} \rho(x=0) &= At^{\eta(\chi-1)(k+1)}(lr)^{-\chi(k+1)} \\ \rho(x=1) &= At^{\eta\chi(k+1)}(lr)^{-(\chi+1)(k+1)} \end{aligned} \tag{18}$$

From here, we can solve for the temperature by two different methods; the first and most obvious is to use the momentum conservation equation (4). With the two different density equations above, we derive a pair of equations for $T(r,t)$:

$$T(x=0) = \frac{\eta - \eta^2}{\Gamma(2-\chi(k+1))}\frac{r^2}{t^2} \tag{19}$$

$$T(x=1) = \frac{\eta - \eta^2}{\Gamma(2-(\chi-1)(k+1)}\frac{r^2}{t^2} \tag{20}$$

Alternatively, we can choose instead to solve the energy conservation equation (4), observing that the non-divergent flux falls out of Coggeshall's original statement:

$$\frac{\Gamma}{\gamma-1}(T_t + uT_r) + \Gamma T u_r + \Gamma T\frac{ku}{r} = 0$$

This leads to a solution for temperature similar in form to our density equation, but with new constants B, y, z, and q to replace A, χ, x, and l, respectively:

$$T(r,t) = \frac{Bt^{\eta y(k+1)(\gamma-1)}}{(z(qr)^{\frac{1}{\eta}} + (1-z)t)^{\eta(k+1)(\gamma-1)}(qr)^{y(k+1)(\gamma-1)}} \tag{21}$$

In order for this equation to be separable, we must likewise impose the requirement that $z = 0$ or $z = 1$. The exponents on r and t must be 2 and -2, respectively, which allows us to solve for some of the newly introduced constants.

$$\begin{aligned} T(z=0) &= Bt^{\eta(y-1)(k+1)(\gamma-1)}(qr)^{-y(k+1)(\gamma-1)} \\ T(z=1) &= Bt^{\eta y(k+1)(\gamma-1)}(qr)^{-(y+1)(k+1)(\gamma-1)} \end{aligned} \tag{22}$$

Reconciling our two equations for temperature yields the following additional requirements:

$$\begin{aligned} \eta &= \frac{2}{2+(k+1)(\gamma-1)} \\ &\begin{cases} \chi(k+1) = \frac{2\beta+k+7}{\alpha} & \text{for } x=0 \\ (\chi+1)(k+1) = \frac{2\beta+k+7}{\alpha} & \text{for } x=1 \end{cases} \end{aligned} \tag{23}$$

Substituting appropriately for χ reduces our density and temperature equations to a single expression each, and we replace the constant expressions in A and l with ρ_0 so as to be compatible with Coggeshall's notation. Substitution of the density and temperature expressions gives us the radiative flux as well. Our full solution, which is the same as Coggeshall's Solution 9 (often referred to as Cog09), is given below.

$$u(r,t) = \left(\frac{2}{2+(k+1)(\gamma-1)}\right)\frac{r}{t} \tag{24}$$

$$\rho(r,t) = \rho_0 t^{-\frac{2}{\alpha}(\frac{\alpha(k+1)-(2\beta+k+7)}{2+(k+1)(\gamma-1)})} r^{-\frac{2\beta+k+7}{\alpha}} \tag{25}$$

$$T(r,t) = \left(\frac{2\alpha(k+1)(\gamma-1)}{\Gamma(2+(k+1)(\gamma-1))^2(2\alpha-(2\beta+k+7))}\right)\frac{r^2}{t^2} \tag{26}$$

$$F(r,t) = \frac{-8ac\lambda_0\rho_0^\alpha}{3\Gamma}\left(\frac{2\alpha(k+1)(\gamma-1)}{(2+(k+1)(\gamma-1))^2(2\alpha-(2\beta+k+7))}\right)^{4+\beta} t^{\frac{2(2\beta+k+7-\alpha(k+1))}{2+(k+1)(\gamma-1)}} r^{-k} \tag{27}$$

The free parameters of this solution are α, β, ρ_0, and the geometric factor k. The constant γ is determined by the properties of the gas.

3 Methods

3.1 Discretization

The solutions described in the previous section yield pointwise values, whereas the output from xRAGE represent data averaged over each cell in the grid. We surmised that we could minimize the differences between the xRAGE and analytic data by comparing the xRAGE data to cell-averaged analytic values. We calculated the cell-averaged analytic data by two methods, which we called the nonconservative and conservative methods.

The nonconservative averaging method is the most straightforward of the two, so we treat that method first. In this case, the desired data quantities (velocity, density, temperature, and pressure) are integrated over each cell and then divided by the cell volume. For a generic variable $x(r)$, the general formula is:

$$\bar{x} = \frac{\int_{r_1}^{r_2} x r^k dr}{\int_{r_1}^{r_2} r^k dr} \tag{28}$$

where $\bar{x}$ denotes the average of the quantity x, and the cell spans the distance r_1 to r_2. This was used to calculate the spatial average of each quantity (u, ρ, T, and P) in the discretized cells.

The fact that mass, momentum, and energy may not be conserved in the preceding case is potentially problematic, because xRAGE was specifically designed to obey conservation laws. We can, however, calculate averages from quantities that are supposed to be conserved, and we begin by computing the cell's mass M, volume V, momentum p, and total energy E:

$$\begin{aligned}
M &= \int_{r_1}^{r_2} \rho r^2 dr \\
V &= \frac{4}{3}\pi(r_2^3 - r_1^3) \\
p &= \int_{r_1}^{r_2} \rho u r^2 dr \\
E &= \int_{r_1}^{r_2} (\frac{1}{2}\rho u^2 + \frac{\Gamma}{\gamma - 1}\rho T) r^2 dr \\
\bar{u} &= \frac{p}{M} \\
\bar{\rho} &= \frac{M}{V} \\
\bar{T} &= \frac{E - \frac{p^2}{2M}}{\frac{\Gamma}{\gamma-1}M} \\
\bar{P} &= \Gamma \bar{\rho} \bar{T}
\end{aligned}$$

The total energy was derived from Newtonian Mechanics and the equation of state:

$$E = \frac{\Gamma}{\gamma - 1} T \tag{29}$$

Averaged distributions obtained by both of these methods were used for comparison with the xRAGE data. As will be discussed in more detail later, they yielded L_1 norms and convergence trends so similar that we finally favored the conservative averaging method alone.

3.2 Classical Heat Conduction Module

Another important detail to note is the fact that all of our solutions, and all of Coggeshall's originally published solutions as well, are derived using an assumption of nondivergent flux. Consequently, our solutions are all independent of flux, but it was noticed that one of the derived solutions (found in Appendix C) and many of Coggeshall's published solutions, including Cog09, are invalid with the inclusion of heat flux, due to a delta function at the origin. As a result, we were only able to test xRAGE's classical heat conduction module with the isothermal solution. The nondivergent flux assumption means, in practical terms, that we do not expect the solution to change regardless of whether there is heat flux present in the system. This allows us to switch the classical heat conduction module on and off in our simulations. Ideally, the output from xRAGE should be the same in either case.

3.3 Simulation Settings and Chosen Parameters

The xRAGE settings and solution parameters for our investigation were chosen independently, with the properties of each individual solution in mind. Both the isothermal solution and Cog09, however, were studied with $\gamma = \frac{5}{3}$ and $k = 2$, to model a polytropic gas in spherical symmetry.

3.3.1 Isothermal Solution

The xRAGE simulations of the isothermal solution were conducted, unless otherwise noted, in a domain of radius 10, with only the portion $0 \leq r < 1.5$ being analyzed to prevent the inclusion of boundary effects, since the boundary effects are not the aim of this work. The constants A and B were set to 1 for simplicity, and Γ was set to $\frac{2}{3}$ so the heat capacity was unity. For the convergence tests, the cell sizes of 0.1, 0.05, 0.025, and 0.0125 were used. Unless otherwise noted, the solution was implemented from time $t = 1$ to $t = 1.5$ and the error convergence calculated at time $t = 1.5$.

3.3.2 Coggeshall 9

For Cog09, we had considerable freedom when choosing α and β. We found it most convenient to select an arbitrary value for α first and then compute the resulting constraints on β. The only mathematically prohibited value of α was 0; according to [1], physically relevant values of α tend to fall within the range $-2 < \alpha < -1$. The value we chose was $\alpha = 1.5$, evidently an unphysical value, which we happened to use during testing. We continued to use it for our investigation because we had developed a sense of the solution behavior for this value.

The constraints on β are both mathematical and physical and are defined by the values of k and α. In order to prevent division by zero or a negative temperature distribution, the following must be true:

$$\begin{cases} \beta < \alpha - \frac{1}{2}(k+7) & \text{for } \alpha > 0 \\ \beta > \alpha - \frac{1}{2}(k+7) & \text{for } \alpha < 0 \end{cases}$$

For $k = 2$ and $\alpha = 1.5$, this reduces to $\beta < -3$. Other variables we can set in Cog09 and the xRAGE input deck are ρ_0, Γ, and the specific heat c_v. For the sake of simplicity, we set $\rho_0 = 1.0$, $\Gamma = \gamma - 1$, and $c_v = 1.0$. Substituting for all parameters except β in Cog09 yields the following

system:

$$
\begin{aligned}
u(r,t) &= \frac{r}{2t} \\
\rho(r,t) &= r^{(-\frac{4}{3}\beta-6)}t^{(\frac{2}{3}\beta+\frac{3}{2})} \\
T(r,t) &= \left(\frac{-1}{2(\beta+3)}\right)\left(\frac{3r}{4t}\right)^2 \\
P(r,t) &= \left(\frac{-3}{16(\beta+3)}\right) r^{(-\frac{4}{3}\beta-4)}t^{(\frac{2}{3}\beta-\frac{1}{2})} \\
S(r,t) &= \sqrt{\frac{-5}{\beta+3}}\left(\frac{r}{4t}\right)
\end{aligned}
$$

When the solutions are seen in this form, the constraint on β is much more apparent. We decided to investigate the solution's behavior with several β values. Noticing that as β approaches the limit -3, the exponent on r approaches -2 in the density equation and 0 in the pressure equation, we decided to select values of β corresponding to integer-value exponents on r in density and pressure. We also wanted a value close to the limit -3. The set we chose was $\beta = \{-3.1, -3.75, -4.5, -5.25, -6.0\}$; the latter four values yield exponents of -1, 0, 1, and 2 in the density equation and 1, 2, 3, and 4 in the pressure equation. We also began preliminary investigations of an additional set of values: $\beta = \{-3.6, -3.8, -3.9, -4.0, -4.1, -4.2, -4.4\}$. This set was chosen because we observed unexpected oscillations in the xRAGE output distributions for late simulation times ($t_{sim} = 1.0$ sec or $t_{sol} = 2.0$ sec) at $\beta = -4.0$ during early test runs. The final simplified Cog09 equations for each β value are given in Appendix E.

All of the Cog09 variations were processed in xRAGE with the same settings. Simulations began at $t_{sol} = 1.0$ sec, saved data at intervals of 0.05 sec, and ran for a total of 1.0 sec. The time step was fixed at $dt = 10^{-5}$ sec, and we captured data for analysis at $t_{sol} = 1.10001$ sec and $t_{sol} = 2.0$ sec. The bulk of our analysis was performed on the data captured at 1.10001 sec, but the later capture time is also important because then the oscillations near $\beta = -4.0$ were readily visible.

The simulations ran on fixed, one-dimensional grids with a minimum radius of 0.0 and a maximum radius of 9.0. Eight different grid spacing sizes were tried: 0.28 (32 cells), 0.14 (64 cells), 0.1 (90 cells), 0.07 (128 cells), 0.05 (180 cells), 0.04 (256 cells), 0.025 (360 cells), and 0.013 (720 cells). Distribution data stored in the input decks and the cell-averaged analytic data for comparison with xRAGE output were double-precision floating-point values, because the xRAGE outputs are returned in double precision.

It should be briefly noted that all simulations for both solutions were run with a fixed boundary condition at the outside edge. As a result, some deviations are expected in the data near the outer boundary, and it is necessary to remove some portion of each distribution, or the "tail." The depth to which such disturbances entered the Cog09 distributions varied strongly with β: errors propagated inward faster for β values nearer to the limit -3. Rather than selecting a single truncation point for all β values by eye, we analyzed the Cog09 data using six truncation points: $r_{max} = \{3.0, 5.0, 7.0, 8.0, 8.5, 9.0\}$.

3.3.3 General Considerations

Other aspects of data processing were the same for both the Redford solution and Cog09. Once xRAGE had finished a given simulation, a program called Hyperion was used to extract the following

variables in text file form: cell centers, density, specific internal energy, temperature, pressure, and velocity. Using the cell centers output by xRAGE, we found the conservative cell averages of the analytic data. Comparing these to the xRAGE output velocity, density, temperature, and pressure, we calculated L_1 norms for each of the grid spacings used. In the case of Cog09 data, we did this for all different β values and tail truncation values.

Once the L_1 norms were calculated for each grid spacing value, we fit them to a power law equation $L_1 = Ax^b$, where L_1 is the L_1 norms data, x is the grid spacing size, and A and b are free parameters in the fit. In this way we obtained a convergence rate b; for each fit, we also computed the coefficient of determination R^2, which provides a quantitative measure for goodness of fit. We considered R^2 values above 0.9 to be indicative of satisfactory correlation.

4 Results

4.1 The Isothermal Solution

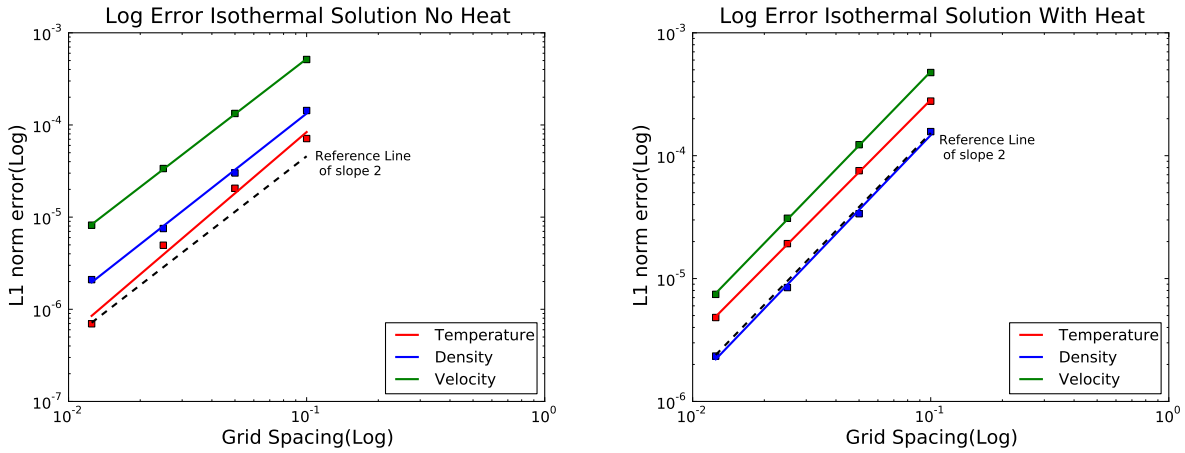


Figure 1: Log-log plot of the grid resolution to error. With and without heat the error exhibits second order convergence.

An actual plot of the results are shown in Figure 2.

The convergence of the solution was approximately second order and exact values are given in table 1.

Another feature worth mentioning in the isothermal solution is that there were small oscillations in the solution that were seen without heat. The cause of these oscillations is currently unknown, but they can be seen upon close inspection of figure 2 and in figure 3.

4.2 Coggeshall 9

Turning our attention to Cog09, we first consider $\beta = \{-3.1, -3.75, -4.5, -5.25, -6.0\}$. When we look at the viewgraph norms in Figure 5, plotting the captured xRAGE data with the averaged analytic data, we find that the plotted lines nearly coincide, as long as we remove the tail of the distribution near the outer boundary. A comparison of the untruncated data at the later capture time in Figure 6 clearly shows how the boundary waves travel farther into the distributions with β closer to -3.

The choice of a truncation point can potentially impact the calculated L_1 norms and con-

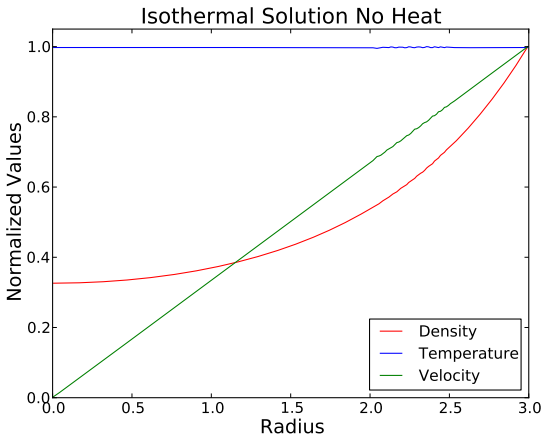


Figure 2: A plot of the xRAGE results without heat normalized so that the maximum value is one so that they can be conveniently plotted on the same graph.

Table 1: Isothermal Solution Convergence

Convervative Discretization Method			
	Temperature	Velocity	Density
No Heat	2.2083	1.9902	2.0289
No Heat R^2	0.9885	0.9999	0.9980
Heat	1.9525	1.9979	2.0199
Heat R^2	0.9998	0.9999	0.9983
Averaging Discretization Method			
	Temperature	Velocity	Density
No Heat	2.1072	1.9893	2.0148
No Heat R^2	0.9965	0.9999	0.9995
Heat	1.9605	2.0114	1.9975
Heat R^2	0.9999	0.9998	0.9995

vergence rates. Shown in Figure 7 is a plot of the L_1 norms for $\beta = -3.1$ and with truncation point $r_{max} = 8.0$. A dashed reference line shows the slope of second-order convergence. Roughly second-order convergence is seen for all quantities except density, which shows roughly first-order convergence. Plots of the L_1 norms for all other β values and most other truncation points are comparable.

We can plot the fitted convergence rate for our set of possible r_{max} values against β to study the convergence trends in parameter space. In Figure 8, different r_{max} values are plotted as colored lines, with β as the independent variable and convergence rate as the dependent variable, for each of the major data quantities of interest. A horizontal reference line lies at a convergence rate of 1.8; above this point, we consider the convergence rate to be close enough to 2 to be called second-order. Likewise, the plots of R^2 have a reference line at 0.9, above which we consider the fit a good one, and below which we are less inclined to trust it. We can remark immediately that r_{max} values of 9.0 (untruncated) and 8.5 give wildly varying values for the convergence rate in most of the plots. We conclude that for these β values at this capture time, we should truncate the distributions to $r_{max} = 8.0$ at the highest. If we look only at convergence data obtained with $r_{max} = 5.0$, as in Figure 9, we see that for all data quantities and β values except density at $\beta = -3.1$, the convergence

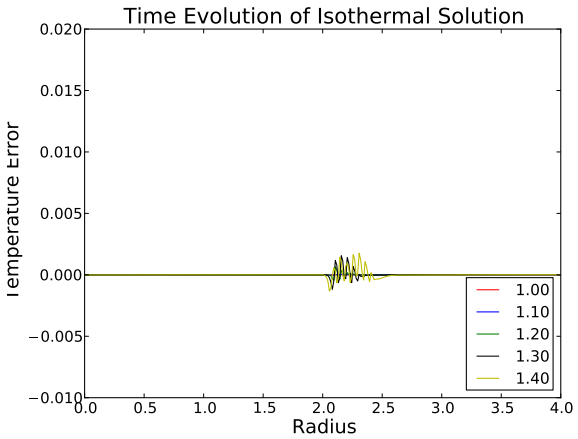


Figure 3: A plot of the error of the xRAGE results from the discretized analytical solution for several times. The starting time was 1. Note that the oscillations are steadily growing with time and spreading but the left side stays at approximately the same position.

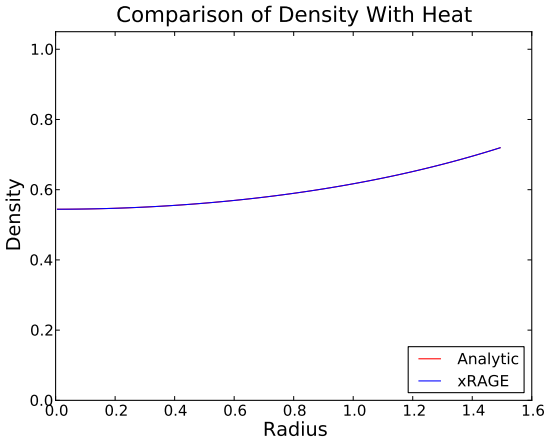


Figure 4: A plot of the analytical solution and xRAGE output on the same graph for easy comparison. Note that the analytical solution is mostly covered by the xRAGE line.

rates are second-order or perhaps a little better. Density at $\beta = -3.1$ shows first-order convergence at best for all r_{max} values. Thus, with an appropriate choice of r_{max}, we observe second-order convergence for nearly all the data quantities with our main group of β values.

If we look at the later capture time $t_{sol} = 2.0$ sec, we find unexpected oscillations in the data for β values near -4.0. Figure 10 shows an example: the xRAGE data without heat is plotted with the conservatively averaged analytic data for $\beta = -4.1$ on a grid with 360 cells. The xRAGE data clearly deviate from the averaged analytic data; what is also interesting is the fact that we do not see similar deviations in the plots for $\beta < -4.5$ or for $\beta > -3.75$. Additional, longer simulation times would need to be run to check whether the range of affected β values increases with simulation length. The oscillations at capture time $t_{sol} = 2.0$ sec seem to peak in amplitude around $\beta = -4.1$ or $\beta = -4.2$.

The amplitude of the oscillations is also affected by the number of cells in the grid, and the impact of the oscillations on the L_1 norm data is striking. Figure 11 shows the L_1 norms versus

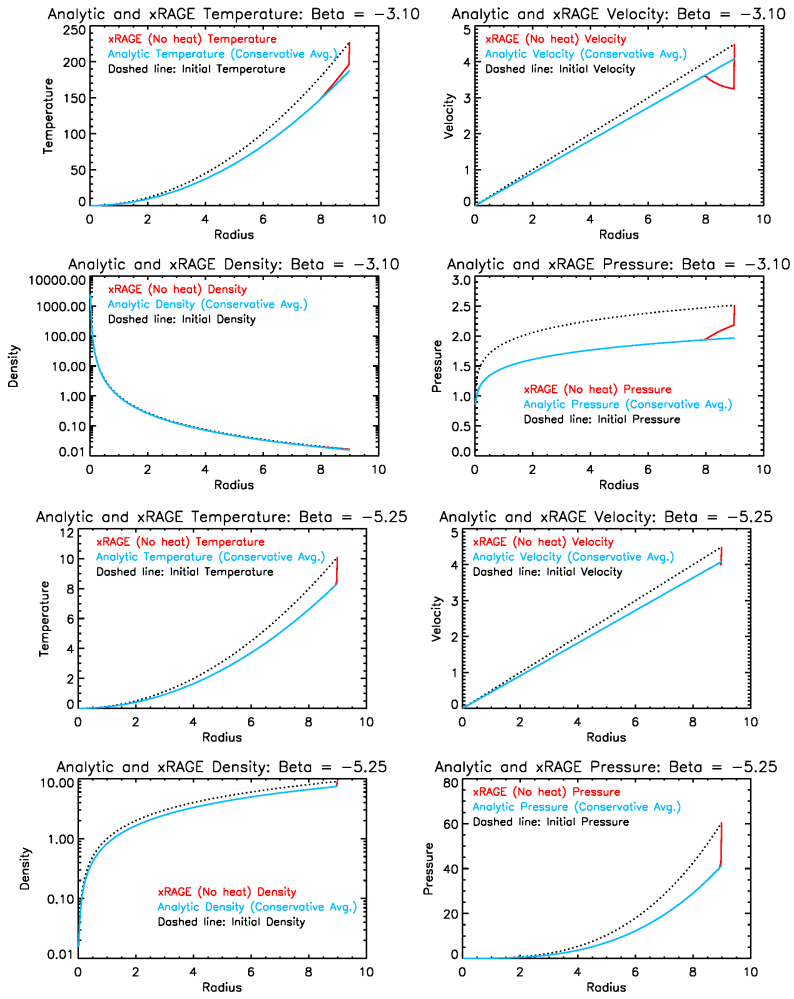


Figure 5: xRAGE distributions compared to conservatively-averaged analytic data at solution time $t_{sol} = 1.10001$ sec. Dashed lines indicate the initial distributions.

grid spacing size for $\beta = -4.1$; notice that although the data appear to show second-order convergence for the largest cell sizes, after 64 cells the convergence rate begins decreasing rapidly until it reaches another peak. As a result, attempting to fit a power law equation to the L_1 norms versus grid spacing in this region is not helpful. The cause of this oscillatory behavior remains unclear.

5 Conclusions

We find in general that both the isothermal solution and Cog09 exhibit the expected second-order convergence when xRAGE is run without the heat module. Because of the delta function at the origin in Cog09 when flux is included, we are only able to report on the convergence of xRAGE data with the heat module on for the isothermal solution.

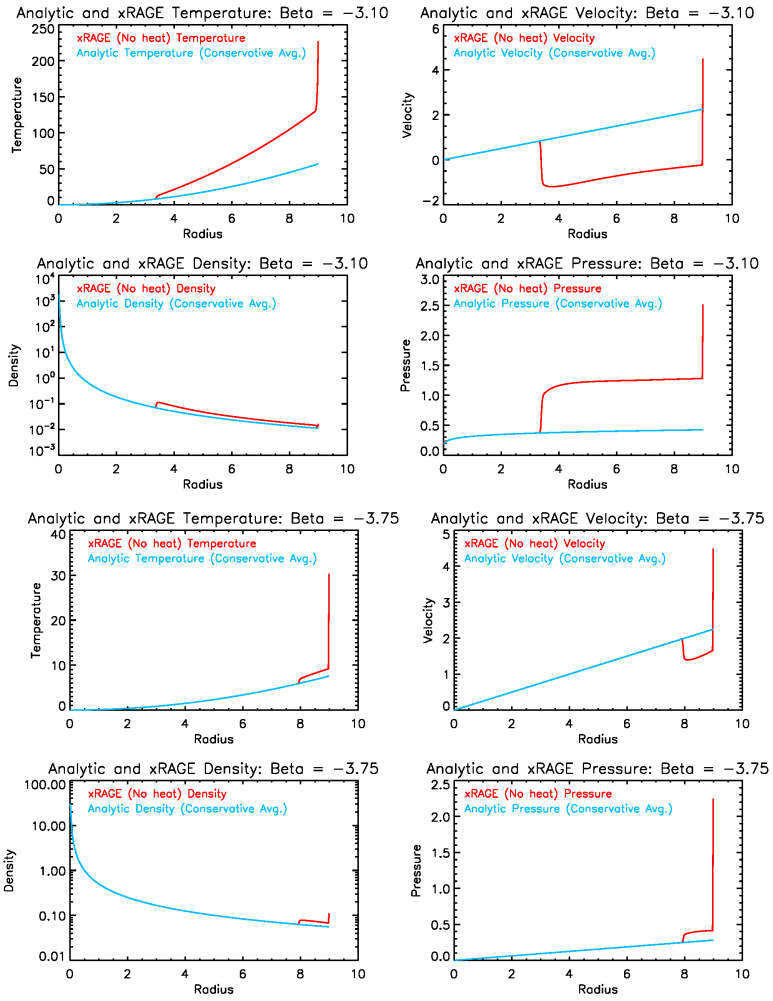


Figure 6: xRAGE distributions compared to conservatively-averaged analytic data at solution time $t_{sol} = 2.0$ sec. Notice that the disturbance from the outer boundary has propagated farther inward for $\beta = -3.10$ than for $\beta = -3.75$.

5.1 Isothermal Solution

The isothermal solution exhibits second order convergence both with and without the heat module. Like Coggeshall 9, it has oscillatory behavior that is currently unexplained. This has been proposed to be a boundary effect [7], but this does seem unusual as it seems to stay fairly stationary, unless this is a standing wave phenomenon.

This test problem has the expected second order convergence with the heat module, but more heat module tests should be used. As will be shown in Appendix C, several past verification efforts including the test problem in [4] and the Coggeshall 9 test problems in [5] are incorrect. Additionally, the isothermal solution exhibits spatially constant temperature, which along with $\alpha = 0$ and $\beta = -3$ should be the simplest case possible for the heat conduction module. So it would be helpful to use more tests on the heat module.

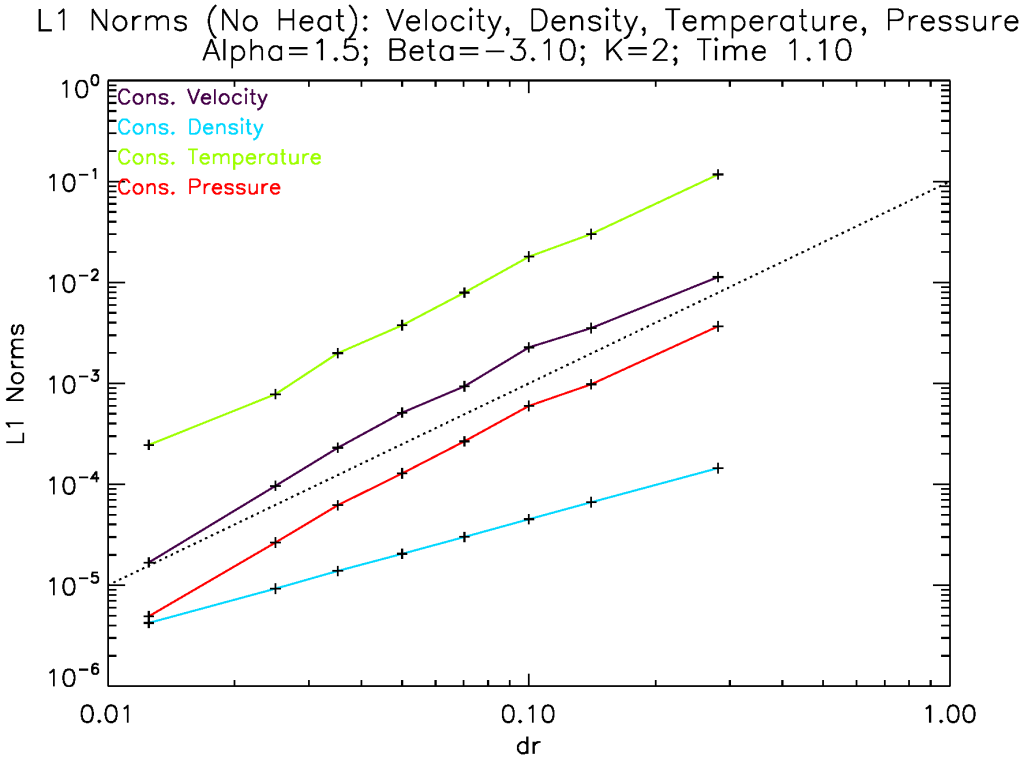


Figure 7: L_1 norms plotted against grid spacing size for $\beta = -3.10$ at solution time $t_{sol} = 1.10001$ sec.

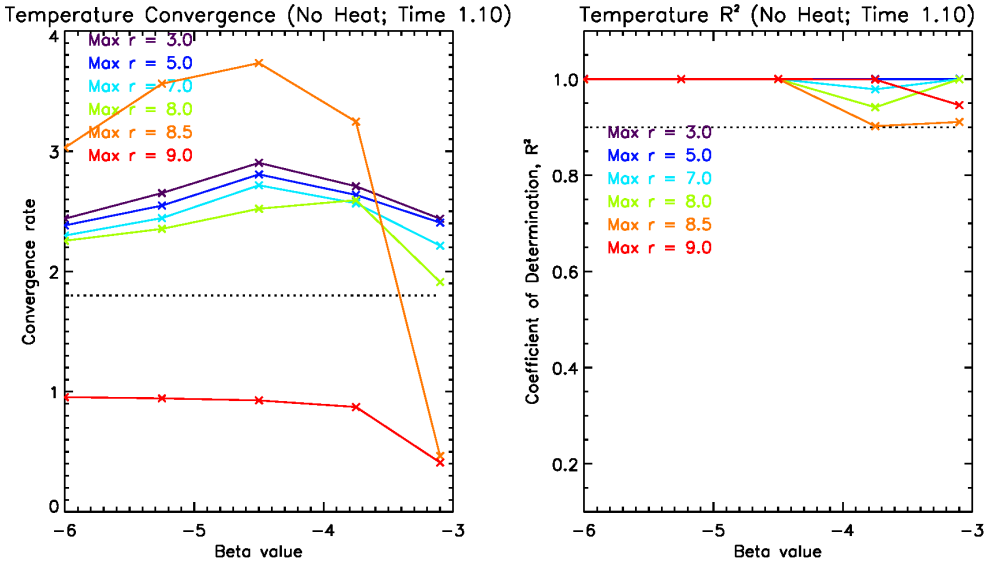


Figure 8: Convergence and R^2 plotted against β for temperature. Each color represents a different value of the truncation point r_{max}.

5.2 Coggeshall 9

For Cog09, we observed no significant difference in convergence trends whether we used conservatively or nonconservatively averaged analytic data for comparison with xRAGE. The trends also

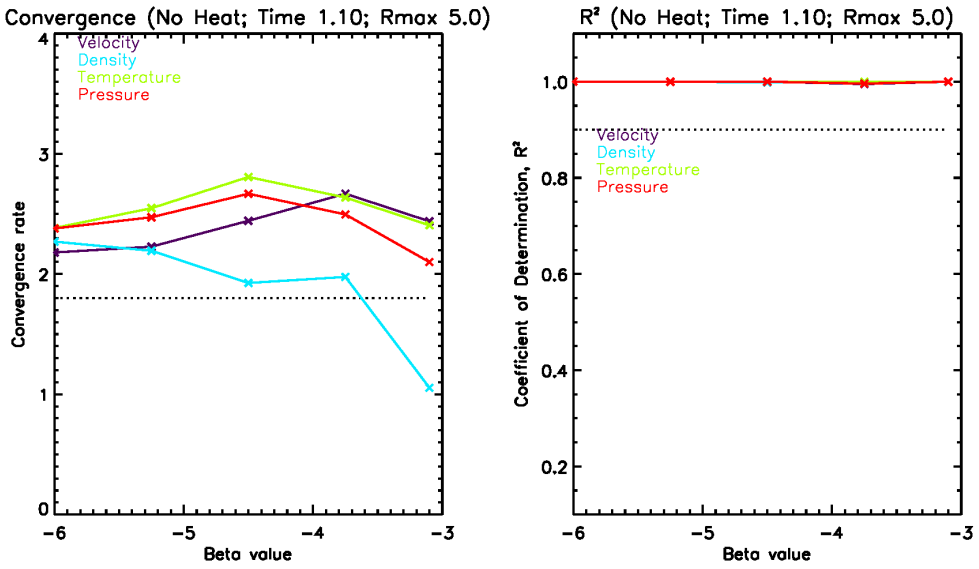


Figure 9: Convergence and R^2 plotted against β for $r_{max} = 5.0$. Each color represents a different data quantity: velocity, density, temperature, or pressure.

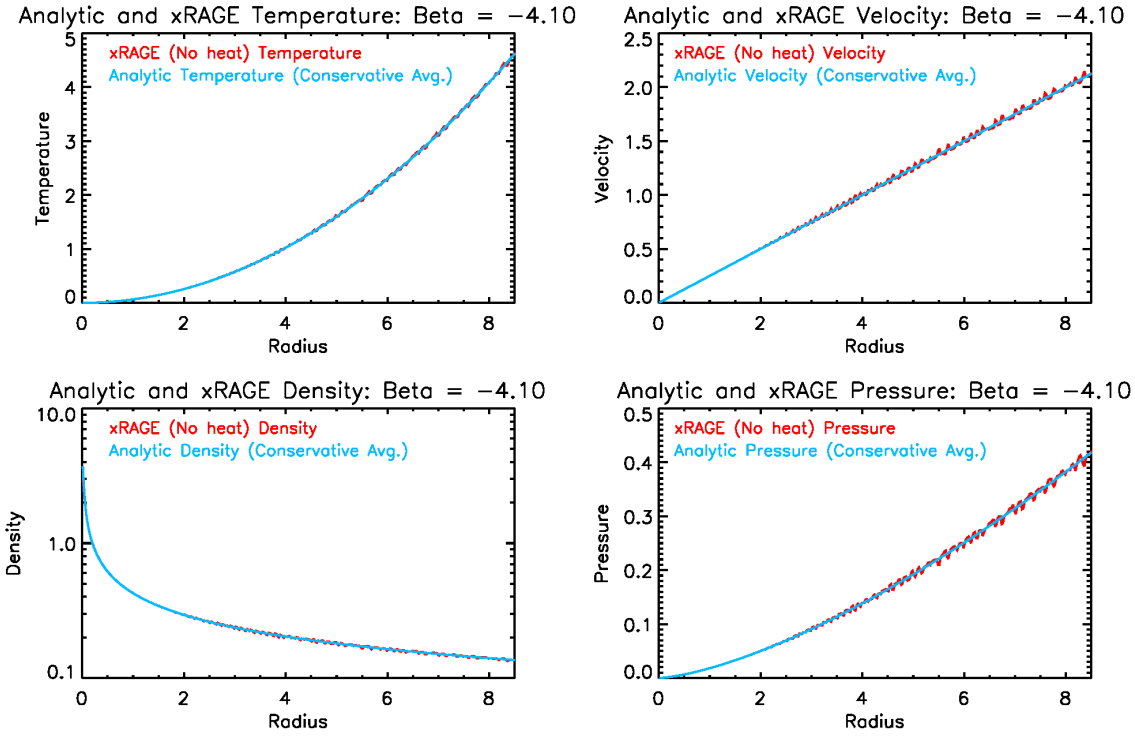


Figure 10: xRAGE distributions plotted with conservatively-averaged analytic solutions for $\beta = -4.10$ at solution time $t_{sol} = 2.0$ sec. Oscillations appear most clearly in velocity and pressure but can also be seen in density and temperature.

remained the same when we compared xRAGE to the pointwise analytic data. As we stated briefly above, second-order convergence with strong correlation was observed for $\beta = \{-3.1, -3.75, -4.5, -5.25, -6.0\}$

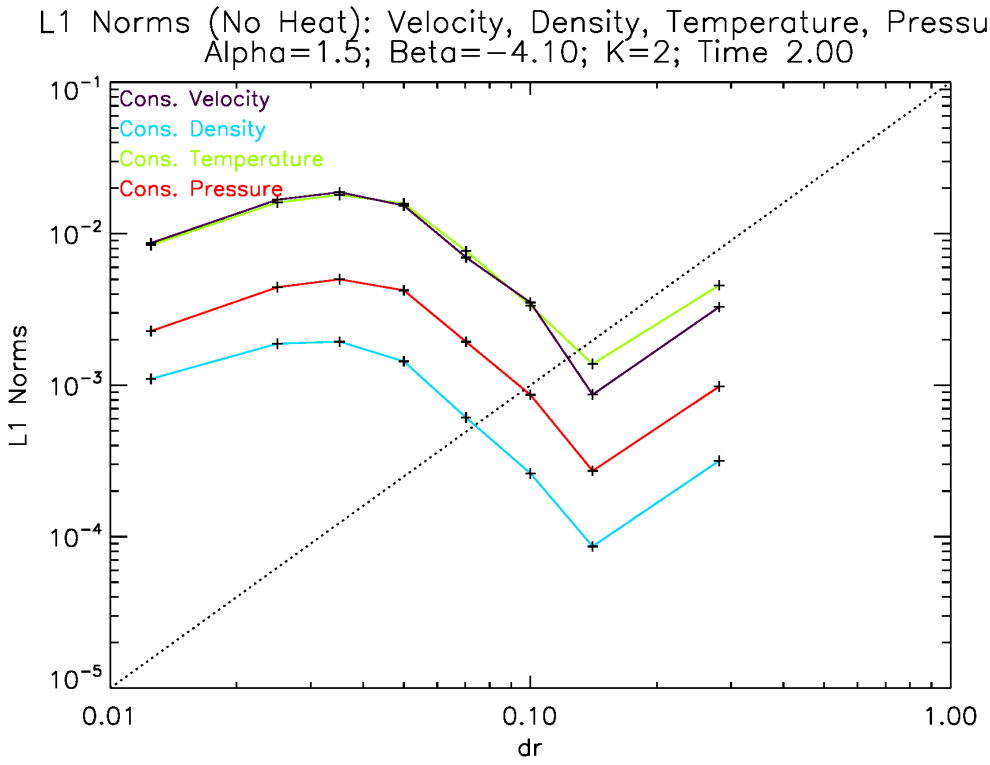


Figure 11: xRAGE distributions plotted with conservatively-averaged analytic solutions for $\beta = -4.10$ at solution time $t_{sol} = 2.0$ sec. Oscillations appear most clearly in velocity and pressure but can also be seen in density and temperature.

for all variables except the density at $\beta = -3.1$. The choice of a truncation point had some impact on the observed convergence, due to boundary errors propagating inward at the sound speed, which is higher for β values nearer the limit -3. In general, we found for this β group that an r_{max} value of 8.0 or lower gave the best results at the capture time $t_{sol} = 1.10001 sec$.

The fact that the sound speed varies with β means that problems with β closer to -3 are inherently more challenging to solve by numerical methods. On the other hand, at later data capture times we observed unexplained oscillations in the xRAGE output distributions when β is near -4.0. At present we can imagine two possible causes or contributing factors:

1.) The oscillations are the result of some special combination of position exponent values in the density and pressure equations. This could then be traced back to some particular relationship between α, β, γ, and k.

2.) The oscillations are standing waves that arise from a lucky combination of the sound speed, grid size, and number of cells.

Different strategies would be needed to probe each possible explanation. In the first case, we would need to select different values of α and/or k and seek the β values that produce similar oscillations in the output data. With enough sets of α, k, and β where such behavior is observed, we could begin to characterize the mathematical relationship that defines when oscillatory behavior arises.

Alternatively, to test the second theory, we should vary β, the grid dimensions, and the grid spacing sizes. It might also help to vary the position of the grid, since sound speed depends not only on β but also on position. Here again the goal would be to understand the ways of combining

β, grid size and position, and grid spacing such that oscillations arise in the output distributions from xRAGE.

One detail remains to be mentioned for our work with Cog09. In the course of our investigation we observed that the results are very sensitive to the precision of the comparisons; for example, when calculating averages of the analytic data for comparison with the xRAGE outputs, the averaging program was initially fed the boundaries of the grid and the number of cells, whence it calculated the cell centers and boundaries itself. As a test, we tried feeding it the cell centers extracted directly from xRAGE instead and made it compute the cell boundaries as the averages of adjacent cell centers. This subtle alteration improved the observed L_1 norms and convergence rates. If we opt to continue comparing the xRAGE data to cell-averages of the analytic data, we may see a similar improvement if we can extract the cell boundaries directly from xRAGE, thus sidestepping the necessity of recalculating cell boundaries in the averaging program. At the moment, the Hyperion program does not have an option for extracting the cell boundary data.

6 Acknowledgements

This work was performed under the auspices of the United States Department of Energy by Los Alamos National Security, LLC, at Los Alamos National Laboratory under contract DE-AC52-06NA25396. The authors acknowledge the support of the US Department of Energy Advanced Strategic Computing Program Computational Physics Student Summer Workshop under University Liaison S. Runnels, and the US Department of Energy Advanced Strategic Computing Program Verification Project under project leader S. Doebling. The authors thank S. Ramsey for valuable insights on these topics, T. Jenkins for technical support, and the University of New Mexico-Los Alamos for use of their facilities.

Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

References

[1] Coggeshall, S., 1991., Analytic solutions of hydrodynamics equations. *Physics of Fluids* A, 3, 757-769.

[2] M. Gittings, et al., "The RAGE radiation-hydrodynamics code," Computational Science & Discovery 1, 015005-015005 (2008).

[3] Pomraning, G. C., *The Equations of Radiation Hydrodynamics*. Dover Publications: Mineola, NY. 2002.

[4] Hendon, R. and Ramsey, S., Radiation Hydrodynamics Test Problems with Linear Velocity Profiles, Los Alamos National Laboratory, 2012.

[5] Marcath, M. and Wang, M., Development and Implementation of Radiation-Hydrodynamics Verification Test Problems, Los Alamos National Laboratory, 2012.

[6] S. Ramsey, personal communication, August 2013.

[7] T. Masser, personal communication, August 2013.

Appendix

A Derivation of Coggeshall's Radiative Hydrodynamics Equations

We began with

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial}{\partial t}(\rho \mathbf{u} + \frac{1}{c^2}\mathbf{F}) + \nabla P_m + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + \mathbf{P_{rad}}) &= 0 \\ \frac{\partial}{\partial t}(\frac{1}{2}\rho u^2 + E_m + E_{rad}) + \nabla \cdot [(\frac{1}{2}\rho u^2 + E_m + P_m)\mathbf{u} + \mathbf{F}] &= 0 \end{aligned} \tag{30}$$

from equations 9.82 to 9.84 of [3]. In these equations, F is the heat flux of the radiation, E_m is the material's internal energy density, E_{rad} is the energy density of the radiation, P_{rad} is radiation pressure, P_m is pressure of the material, u is the fluid velocity, and ρ is the fluid density. With Coggeshall's first assumption of a one-dimensional geometry we could simplify the gradients as shown:

$$\begin{aligned} \nabla f &= \frac{\partial f}{\partial x}\hat{\mathbf{x}} + \frac{\partial f}{\partial y}\hat{\mathbf{y}} + \frac{\partial f}{\partial z}\hat{\mathbf{z}} = \frac{\partial f}{\partial x}\hat{\mathbf{x}} \text{ in Cartesian} \\ &= \frac{\partial f}{\partial \rho}\hat{\rho} + \frac{1}{\rho}\frac{\partial f}{\partial \phi}\hat{\phi} + \frac{\partial f}{\partial z}\hat{\mathbf{z}} = \frac{\partial f}{\partial \rho}\hat{\rho} \text{ in Cylindrical} \\ &= \frac{\partial f}{\partial r}\hat{\mathbf{r}} + \frac{1}{r}\frac{\partial f}{\partial \theta}\hat{\theta} + \frac{1}{r\sin\theta}\frac{\partial f}{\partial \phi}\hat{\phi} = \frac{\partial f}{\partial r}\hat{\mathbf{r}} \text{ in Spherical} \end{aligned}$$

We used r for the spatial coordinate, so for all geometries considered $\nabla f = \frac{\partial f}{\partial r}$. Similarly, for the divergence

$$\begin{aligned} \nabla \cdot \mathbf{f} &= \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y} + \frac{\partial f_z}{\partial z} = \frac{\partial f_x}{\partial x} = \frac{\partial f}{\partial x} \text{ in Cartesian} \\ &= \frac{1}{\rho}\frac{\partial(\rho f_\rho)}{\partial \rho} + \frac{1}{\rho}\frac{\partial f_\phi}{\partial \phi} + \frac{\partial f_z}{\partial z} = \frac{1}{\rho}\frac{\partial(\rho f_\rho)}{\partial \rho} \\ &= \frac{1}{\rho}(f_\rho + \rho\frac{\partial f_\rho}{\partial \rho}) = \frac{\partial f_\rho}{\partial \rho} + \frac{f_\rho}{\rho} = \frac{\partial f}{\partial \rho} + \frac{f}{\rho} \text{ in Cylindrical} \\ &= \frac{1}{r^2}\frac{\partial(r^2 f_r)}{\partial r} + \frac{1}{r\sin\theta}\frac{\partial f_\theta \sin\theta}{\partial \theta} + \frac{1}{r\sin\theta}\frac{\partial f_\phi}{\partial \phi} = \frac{1}{r^2}\frac{\partial r^2 f_r}{\partial r} \\ &= \frac{1}{r^2}(r^2\frac{\partial f_r}{\partial r} + 2r f_r) = \frac{\partial f_r}{\partial r} + \frac{2f_r}{r} = \frac{\partial f}{\partial r} + \frac{2f}{r} \text{ in Spherical} \end{aligned}$$

This can be summarized as $\nabla \cdot f = \frac{\partial f}{\partial r} + \frac{kf}{r}$ where k is a geometrical factor that is 0 in Cartesian, 1 in cylindrical, and 2 in spherical coordinates. Starting with the first equation in (30), we found

$$\begin{aligned} 0 &= \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ &= \frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial r} + \frac{k\rho u}{r} \\ &= \frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial r} + \rho\frac{\partial u}{\partial r} + \frac{k\rho u}{r} \\ &= \rho_t + u\rho_r + \rho u_r + \frac{k\rho u}{r} \text{ using subscript notation} \end{aligned}$$

which is equation (1a) in [1]. The second equation became

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}(\rho\mathbf{u} + \frac{1}{c^2}\mathbf{F}) + \nabla P_m + \nabla \cdot (\rho\mathbf{u}\mathbf{u} + \mathbf{P_{rad}}) \\ &= \frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla P + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) \end{aligned}$$

using the approximation that the pressure and momentum of the fluid is much greater than that of the radiation. Then we eliminated P by using the ideal gas equation of state

$$P = \Gamma\rho T \tag{31}$$

where Γ is the gas constant and T is the temperature.

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla(\Gamma\rho T) + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) \\ &= \rho_t u + \rho u_t + \Gamma\rho_r T + \Gamma\rho T_r + \frac{\partial}{\partial r}(\rho u^2) + \frac{k\rho u^2}{r} \\ &= \rho_t u + \frac{k\rho u^2}{r} + \rho u_t + \Gamma\rho_r T + \Gamma\rho T_r + 2uu_r\rho + \rho_r u^2 \\ &= u(\rho_t + \rho u_r + u\rho_r + \frac{k\rho u}{r}) + \rho u_t + \rho u u_r + \Gamma\rho_r T + \Gamma\rho T_r \\ &= u(0) + \rho u_t + \rho u u_r + \Gamma\rho_r T + \Gamma\rho T_r \\ &= u_t + u u_r + \Gamma\rho_r T + \Gamma\rho T_r \end{aligned}$$

which is equation (1b) of [1]. Finally, we went to the third equation of (30),

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}(\frac{1}{2}\rho u^2 + E_m + E_{rad}) + \nabla \cdot [(\frac{1}{2}\rho u^2 + E_m + P_m)\mathbf{u} + \mathbf{F}] \\ &= \frac{\partial}{\partial t}(\frac{1}{2}\rho u^2 + E) + \nabla \cdot [(\frac{1}{2}\rho u^2 + E + P)\mathbf{u} + \mathbf{F}] \end{aligned}$$

where we have assumed that the internal energy of the fluid is much greater than the radiation energy. Then we substituted (31) for P as before. The internal energy equation for an gamma law ideal gas provided us with an expression for the internal energy density E.

$$E = \frac{\Gamma}{\gamma - 1}\rho T \tag{32}$$

where γ is the adiabatic index.

$$
\begin{aligned}
0 &= \frac{\partial}{\partial t}(\frac{1}{2}\rho u^2 + \frac{\Gamma}{\gamma-1}\rho T) + \frac{\partial}{\partial r}((\frac{1}{2}\rho u^2 + \frac{\Gamma}{\gamma-1}\rho T + \Gamma\rho T)u) + \frac{(\frac{1}{2}\rho u^2 + \frac{\Gamma}{\gamma-1}\rho T + \Gamma\rho T)ku}{r} + \frac{\partial F}{\partial r} + \frac{kF}{r} \\
&= \rho u u_t + \frac{1}{2}u^2\rho_t + \frac{\Gamma}{\gamma-1}\rho_t T + \frac{\Gamma}{\gamma-1}\rho T_t + \frac{1}{2}\rho_r u^3 + \frac{3}{2}u^2 u_r \rho + \frac{\Gamma}{\gamma-1}\rho_r u T + \frac{\Gamma}{\gamma-1}\rho T_r u + \Gamma\rho T_r u + \Gamma\rho_r T u \\
&\quad + (\frac{\Gamma}{\gamma-1}\rho T + \Gamma\rho T)u_r + \frac{(\frac{1}{2}\rho u^2 + \frac{\Gamma}{\gamma-1}\rho T + \Gamma\rho T)ku}{r} + F_r + \frac{kF}{r} \\
&= \frac{1}{2}u^2(\rho_t + u\rho_r + u_r\rho + \frac{k\rho u}{r}) + \rho u(u_t + u u_r + \frac{1}{\rho}\Gamma T\rho_r + \Gamma T_r) + \frac{\Gamma}{\gamma-1}T(\rho_t + u\rho_r + \rho u_r + \frac{k\rho u}{r}) \\
&\quad + \frac{\Gamma}{\gamma-1}(\rho T_t + \rho T_r u) + \Gamma\rho u_r T + \Gamma T\frac{ku}{r} + F_r + \frac{kF}{r} \\
&= \frac{1}{2}u^2(0) + \rho u(0) + \frac{\Gamma}{\gamma-1}T(0) + \frac{\Gamma}{\gamma-1}(\rho T_t + \rho T_r u) + \Gamma\rho u_r T + \Gamma T\frac{k\rho u}{r} + F_r + \frac{kF}{r} \\
&= \frac{\Gamma\rho}{\gamma-1}(T_t + uT_r) + \rho\Gamma T u_r + \Gamma T\rho\frac{ku}{r} + \frac{\rho}{\rho}(F_r + \frac{kF}{r}) \\
&= \frac{\Gamma}{\gamma-1}(T_t + uT_r) + \Gamma T u_r + \Gamma T\frac{ku}{r} + \frac{1}{\rho}(F_r + \frac{kF}{r})
\end{aligned}
$$

which is equation (1c) of [1].

B Full Derivation of Solutions

We began with

$$
\begin{aligned}
T &= g(t) \\
\rho &= \frac{h(\frac{r}{t^\eta})}{t^{\eta(k+1)}}
\end{aligned}
$$

$$
\begin{aligned}
T &= g(t) \\
0 &= u_t + u u_r + \frac{\rho_r}{\rho}\Gamma T + \Gamma T_r \\
0 &= \frac{(\eta^2-\eta)r}{t^2} + \frac{\rho_r}{\rho}\Gamma g(t) \\
\frac{\rho_r}{\rho} &= -\frac{(\eta^2-\eta)r}{\Gamma g(t)t^2} \\
ln(\rho) &= -\frac{(\eta^2-\eta)r^2}{2\Gamma g(t)t^2} + h(t) \\
\rho &= e^{h(t)}Exp[-\frac{(\eta^2-\eta)r^2}{2\Gamma g(t)t^2}]
\end{aligned}
$$

For this to have the form $\rho = \frac{j(\frac{r}{t^\eta})}{t^{\eta(k+1)}}$, we required

$$\begin{aligned}
e^{h(t)} &= At^{-\eta(k+1)} \\
\frac{r^2}{g(t)t^2} &\propto \frac{r^2}{t^{2\eta}} \\
g(t)t^2 &\propto t^{2\eta} \\
g(t) &= Bt^{2\eta-2} \\
\rho &= \frac{A}{t^{\eta(k+1)}} Exp[-\frac{(\eta^2-\eta)r^2}{2\Gamma Bt^{2\eta}}] \\
T &= Bt^{2\eta-2}
\end{aligned}$$

We then returned to equation (4)

$$\begin{aligned}
0 &= \frac{\Gamma}{\gamma-1}(T_t + uT_r) + \Gamma T u_r + \Gamma T \frac{ku}{r} \\
0 &= T_t + uT_r + (\gamma-1)(u_r + \frac{ku}{r})T \\
0 &= (2\eta-2)Bt^{2\eta-3} + 0 + (\gamma-1)\frac{(k+1)\eta}{t} Bt^{2\eta-2} \\
0 &= (2\eta-2) + (\gamma-1)(k+1)\eta \\
2 &= 2\eta + (\gamma-1)(k+1)\eta \\
2 &= \eta(2 + (\gamma-1)(k+1)) \\
\eta &= \frac{2}{2+(\gamma-1)(k+1)}
\end{aligned}$$

So the isothermal solution is

$$\begin{aligned}
\rho &= \frac{A}{t^{\eta(k+1)}} Exp[-\frac{(\eta^2-\eta)r^2}{2\Gamma Bt^{2\eta}}] \\
T &= Bt^{2\eta-2} \\
u &= \frac{\eta r}{t} \\
\eta &= \frac{2}{2+(\gamma-1)(k+1)}
\end{aligned}$$

A second solution was derived, but was later found to be valid only in the case of $k = 0$, as discussed in the next section. Starting from

$$\begin{aligned}
T &= \frac{f(t)}{r^{k-1}} + g(t) \\
\rho &= \frac{h(\frac{r}{t^\eta})}{t^{\eta(k+1)}}
\end{aligned}$$

And then setting $g(t) = 0$

$$\begin{aligned}
0 &= u_t + uu_r + \frac{1}{\rho}\Gamma T\rho_r + \Gamma T_r \\
&= -\frac{\eta r}{t^2} + \frac{\eta^2 r}{t^2} + \frac{1}{\rho}\Gamma\frac{f(t)}{r^{k-1}}\rho_r + \Gamma(1-k)\frac{f(t)}{r^{k+1}} \\
\frac{\rho_r}{\rho}(\frac{f(t)}{r^{k-1}}) + (1-k)\frac{f(t)}{r^k} &= -\frac{(\eta^2-\eta)r}{\Gamma t^2} \\
\frac{\rho_r}{\rho} &= \frac{(k-1)}{r} - \frac{(\eta^2-\eta)r^k}{\Gamma f(t)t^2} \\
ln(\rho) &= ln(r^{k-1}) - \frac{1}{k+1}\frac{(\eta^2-\eta)r^{k+1}}{f(t)\Gamma t^2} + h(t) \\
\rho &= r^{k-1}e^{h(t)}Exp[-\frac{(\eta^2-\eta)r^{k+1}}{f(t)(k+1)\Gamma t^2}]
\end{aligned}$$

From here we assumed $k \neq 1$. To be of the form $\rho = \frac{j(\frac{r}{t^\eta})}{t^{\eta(k+1)}}$ where $j(t)$ is an arbitrary function of t, then $e^{h(t)} = Bt^{2\eta k}$ and

$$\begin{aligned}
\frac{r^{k+1}}{f(t)t^2} &\propto (\frac{r}{t^\eta})^{k+1} \\
f(t)t^2 &\propto t^{\eta(k+1)} \\
f(t) &\propto t^{\eta(k+1)-2} \\
f(t) &= At^{\eta(k+1)-2}
\end{aligned}$$

$$\begin{aligned}
\rho &= B\frac{r^{k-1}}{t^{2\eta k}}Exp[-\frac{(\eta^2-\eta)r^{k+1}}{A(k+1)\Gamma t^{\eta(k+1)}}] \\
T &= A\frac{t^{\eta(k+1)-2}}{r^{k-1}}
\end{aligned}$$

We now applied this to equation (4)

$$0 = \frac{\Gamma}{\gamma - 1}(T_t + uT_r) + \Gamma T u_r + \Gamma T \frac{ku}{r}$$
$$0 = T_t + \frac{\eta r}{t} T_r + \frac{(k+1)\eta(\gamma-1)}{t} T$$

$$T_t = (\eta(k+1) - 2) A \frac{t^{\eta(k+1)-3}}{r^{k-1}}$$
$$T_r = (1-k) A \frac{t^{\eta(k+1)-2}}{r^k}$$
$$0 = (\eta(k+1) - 2) A \frac{t^{\eta(k+1)-3}}{r^{k-1}} + \frac{\eta r}{t}(1-k) A \frac{t^{\eta(k+1)-2}}{r^k} + \frac{(k+1)\eta(\gamma-1)}{t} A \frac{t^{\eta(k+1)-2}}{r^{k-1}}$$
$$0 = \eta(k+1) - 2 + \eta(1-k) + (k+1)\eta(\gamma-1)$$
$$\frac{2}{\eta} = k + 1 + 1 - k + (k+1)(\gamma-1)$$
$$\eta = \frac{2}{2 + (k+1)(\gamma-1)}$$

So the first solution is

$$\rho = B \frac{r^{k-1}}{t^{2\eta k}} Exp[-\frac{(\eta^2 - \eta) r^{k+1}}{A(k+1)\Gamma t^{\eta(k+1)}}]$$
$$T = A \frac{t^{\eta(k+1)-2}}{r^{k-1}}$$
$$u = \frac{\eta r}{t}$$
$$\eta = \frac{2}{2 + (k+1)(\gamma-1)}$$

C Of Coggeshall and Origins

During the testing of the second new "solution"

$$\rho = B \frac{r^{k-1}}{t^{2\eta k}} Exp[-\frac{(\eta^2 - \eta) r^{k+1}}{A(k+1)\Gamma t^{\eta(k+1)}}]$$
$$T = A \frac{t^{\eta(k+1)-2}}{r^{k-1}}$$
$$u = \frac{\eta r}{t} \tag{33}$$
$$\eta = \frac{2}{2 + (\gamma-1)(k+1)}$$
$$k \neq 1$$

It was observed that the equation (4) was not satisfied at the origin. It was derived so that the heat flux term in (4) was zero with $\alpha = 0$ and $\beta = -3$. But if attention is paid to the flux term, it will be noticed that it is, in fact $\nabla \cdot F$. With $\alpha = 0$ and $\beta = -3$ then the flux is $F \propto \nabla T$. Therefore

the flux term becomes ΔT, which means that for $k = 2$(spherical geometry)

$$\begin{aligned}\Delta T &= \Delta(A\frac{t^{\eta 3-2}}{r}) \\ &= At^{\eta 3-2}\Delta(\frac{1}{r}) \\ &= At^{\eta 3-2}4\pi\delta(r) \\ &\neq 0\end{aligned} \tag{34}$$

So the solution will fail at the origin. When this was implemented in xRAGE the results deviated from the analytical solution at the origin. To illustrate this, some graphs of the analytical solution and xRAGE output is shown in figure 12.

It can be qualitatively seen that with heat conduction the solution cools at the origin which causes

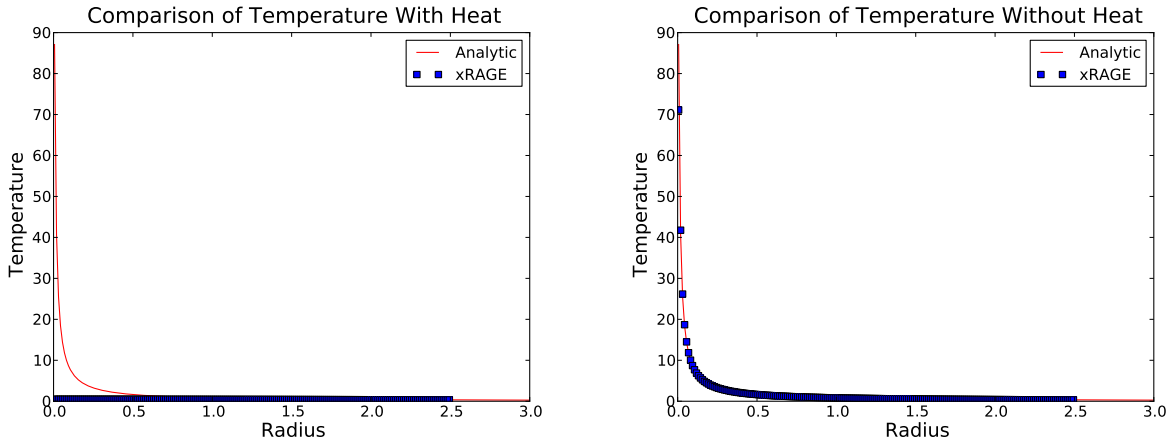


Figure 12: Left is the plot of xRAGE output for heat conduction(left) and no heat conduction(right), plotted on the same axes as the analytical solution. Note that (33) is a valid solution only when the heat conduction term is ignored(no heat conduction).

a decrease in pressure and material collapses into the origin. After realizing that the solution fails at the origin as shown in (34), it was decided to investigate the same problem with Coggeshall's solutions. First looking at the divergence of $r^{-k}\hat{r}$ using the divergence theorem

$$\oint\oint\oint \nabla\cdot(r^{-k}\hat{r})dV = \oint\oint r^{-k}\hat{r}\cdot dS$$

Integrating over a sphere($k = 2$)/cylinder($k = 1$)/cube($k = 0$) of radius/side length R

$$\oint\oint\oint_0^R \nabla\cdot(r^{-k}\hat{r})dV = CR^{-k}R^k$$

where $C = 0$ for $k = 0$, $C = 2\pi z$ for $k = 1$, and $C = 4\pi$ for $k = 2$

$$\oint\oint\oint_0^R \nabla\cdot(r^{-k}\hat{r})dV = C \tag{35}$$

which is independent of the radius. Therefore

$$\lim_{R\to 0}\oint\oint\oint_0^R \nabla\cdot(r^{-k}\hat{r})dV = C \tag{36}$$

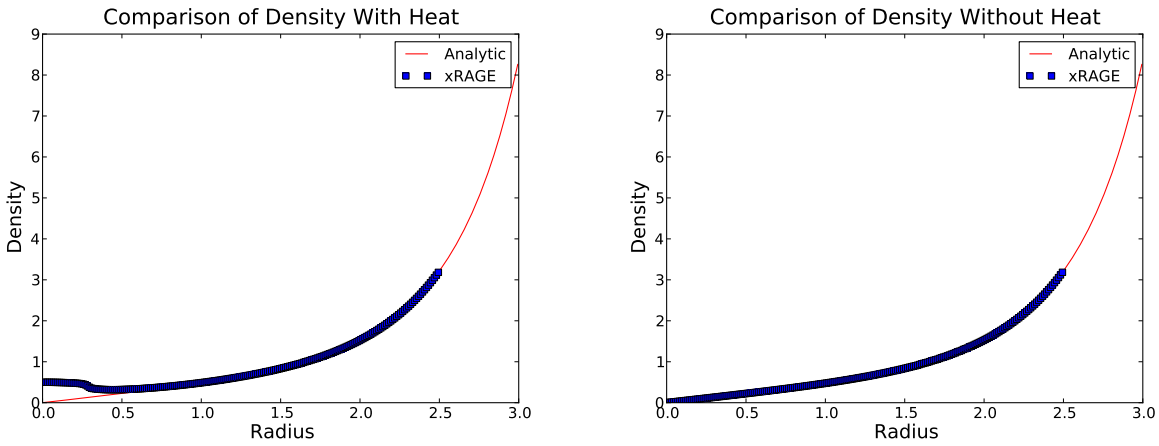


Figure 13: Similar to figure 12, this figure shows density compared between xRAGE with heat conduction(left) and without heat conduction(right). The analytical solution is plotted on both graphs for comparison.

And

$$\begin{aligned}\oint\oint\oint_{R_1}^{R_2} \nabla\cdot(r^{-k}\hat{r})dV &= \oint\oint\oint_{0}^{R_2} \nabla\cdot(r^{-k}\hat{r})dV - \oint\oint\oint_{0}^{R_1} \nabla\cdot(r^{-k}\hat{r})dV \\ &= C - C \\ &= 0\end{aligned} \tag{37}$$

So the volume is C if the region contains the origin and zero otherwise. So also taking into account the values of C

$$\nabla\cdot(r^{-k}\hat{r}) = 2k\pi\delta(r) \tag{38}$$

Now going back to the derivation of (2), It will be noted that (2) is just the simplified form of

$$\rho_t + \nabla\cdot(\rho\mathbf{u}) = 0$$

which means that from the previous equations if $\rho\mathbf{u}$ is proportional to r^{-k} there will be a delta function in non-Cartesian geometries. Looking over Coggeshall's solutions there are several solutions with this form.(Note, that unless explicitly forbidden the solutions are still valid for $k = 0$)

Coggeshall 4 and 12

$$\begin{aligned}\rho &= \rho_0 r^{-2k/(\gamma+1)} \\ (u) &= u_0 r^{-k(\gamma-1)/(\gamma-1)}\hat{r}\end{aligned}$$

$$\begin{aligned}\rho_t + \nabla\cdot(\rho\mathbf{u}) &= 0 \\ &= 0 + \nabla\cdot(\rho_0 r^{-2k/(\gamma+1)} u_0 r^{-k(\gamma-1)/(\gamma-1)}\hat{r}) \\ &= \nabla\cdot(\rho_0 u_0 r^{-k(\gamma+1)/(\gamma+1)}\hat{r}) \\ &= \rho_0 u_0 \nabla\cdot(r^{-k}\hat{r}) \\ &= \rho_0 u_0 k 2\pi\delta(r) \neq 0\end{aligned}$$

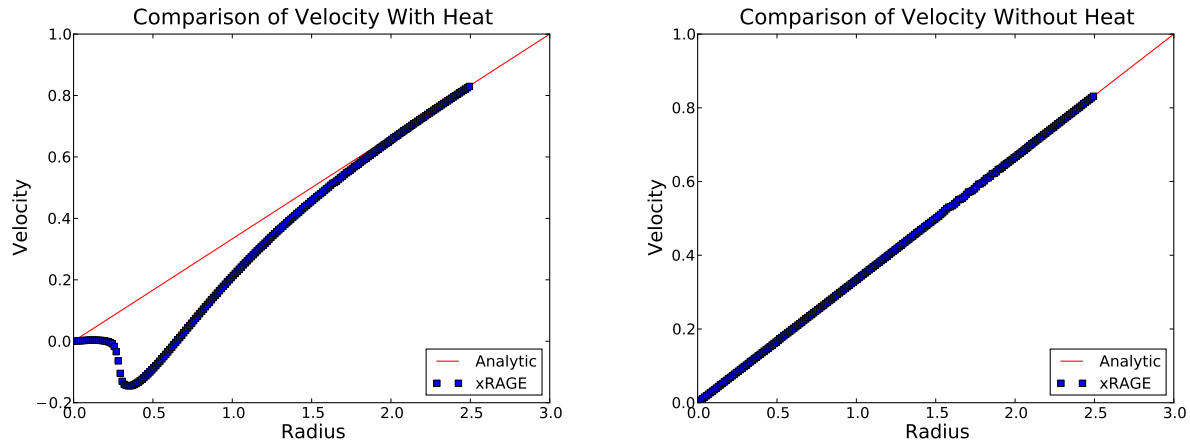


Figure 14: Similar to figures 12 and 13, this figure shows velocity compared between xRAGE with heat conduction(left) and without heat conduction(right). The analytical solution is plotted on both graphs for comparison.

Coggeshall 5

$$
\begin{aligned}
k &= 2 \text{ the only allowed k value for this solution} \\
\rho &= \rho_0 r^{-2} \\
\mathbf{u} &= u_0 t \hat{r}
\end{aligned}
$$

$$
\begin{aligned}
\rho_t + \nabla \cdot (\rho \mathbf{u}) &= 0 + \nabla \cdot (\rho_0 r^{-2} u_0 t \hat{r}) \\
&= u_0 \rho_0 \nabla \cdot (r^{-2} \hat{r}) \\
&= u_0 \rho_0 (4\pi \delta(r)) \neq 0
\end{aligned}
$$

Coggeshall 10

$$
\begin{aligned}
\rho &= \rho_0 r^{-k} \\
\mathbf{u} &= (4c\lambda_0 a/3)[(\gamma - 1)/(\Gamma\gamma)] k \rho_0^{\alpha-1} T_0^{\beta+3} \hat{r}
\end{aligned}
$$

$$
\begin{aligned}
\rho_t + \nabla \cdot (\rho \mathbf{u}) &= 0 + \nabla \cdot (\rho_0 r^{-k} (4c\lambda_0 a/3)[(\gamma - 1)/(\Gamma\gamma)] k \rho_0^{\alpha-1} T_0^{\beta+3} \hat{r}) \\
&= (4c\lambda_0 a/3)[(\gamma - 1)/(\Gamma\gamma)] k \rho_0^{\alpha-1} T_0^{\beta+3} \rho_0 \nabla \cdot (r^{-k} \hat{r}) \\
&= (4c\lambda_0 a/3)[(\gamma - 1)/(\Gamma\gamma)] k \rho_0^{\alpha-1} T_0^{\beta+3} \rho_0 (k 2\pi \delta(r)) \neq 0
\end{aligned}
$$

Coggeshall 14

$$\rho = \rho_0 r^{-k-b}$$
$$\mathbf{u} = r^b \sqrt{\Gamma T_0 (k-b)/b}\hat{r}$$

$$\begin{aligned}\rho_t + \nabla \cdot (\rho \mathbf{u}) &= 0 + \nabla \cdot (\rho_0 r^{-k-b} r^b \sqrt{\Gamma T_0 (k-b)/b}\hat{r}) \\ &= \rho_0 \sqrt{\Gamma T_0 (k-b)/b} \nabla \cdot (r^{-k}\hat{r}) \\ &= \rho_0 \sqrt{\Gamma T_0 (k-b)/b} (k 2\pi \delta(r)) \neq 0\end{aligned}$$

Coggeshall 16

$$\rho = \rho_0 r^{-k-b}$$
$$\mathbf{u} = u_0 r^b \hat{r}$$

$$\begin{aligned}\rho_t + \nabla \cdot (\rho \mathbf{u}) &= 0 + \nabla \cdot (\rho_0 r^{-k-b} u_0 r^b \hat{r}) \\ &= \rho_0 u_0 \nabla \cdot (r^{-k}\hat{r}) \\ &= \rho_0 u_0 (k 2\pi \delta(r)) \neq 0\end{aligned}$$

Coggeshall 22

Coggeshall 22 has two regions and a shock, but the region that contains the origin has ρ and $\mathbf{u}$ identical to Coggeshall 4 and 12.

Now looking at the Conservation of Energy equation (equation (4)), the heat flux term takes the form of

$$\nabla \cdot \mathbf{F} = \nabla \cdot (-\frac{4ac\lambda_0}{3} \rho^\alpha T^{\beta+3}) \nabla T \tag{39}$$

Now looking at several particular equations

Coggeshall 8

$$\rho = \rho_0 r^{\frac{k-1}{\beta-\alpha+4}} t^{-k-1-\frac{k-1}{\beta-\alpha+4}}$$
$$T = T_0 r^{\frac{1-k}{\beta-\alpha+4}} t^{(1-\gamma)(k+1)+\frac{k-1}{\beta-\alpha+4}}$$

$$\nabla T = T_0 \frac{1-k}{\beta-\alpha+4} r^{\frac{1-k-\beta+\alpha-4}{\beta-\alpha+4}} t^{(1-\gamma)(k+1)+\frac{k-1}{\beta-\alpha+4}} \hat{r}$$
$$-\frac{4ac\lambda_0}{3}\rho^\alpha T^{\beta+3}\nabla T = -\frac{4ac\lambda_0}{3}\rho_0^\alpha r^{\frac{\alpha(k-1)}{\beta-\alpha+4}} t^{\alpha(-k-1-\frac{k-1}{\beta-\alpha+4})} T_0^{\beta+4} t^{(\beta+4)((1-\gamma)(k+1)+\frac{k-1}{\beta-\alpha+4})} r^{(\beta+3)(\frac{1-k}{\beta-\alpha+4})} r^{\frac{1-k-\beta+\alpha-4}{\beta-\alpha+4}} \hat{r}$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{\frac{\alpha(k-1)+(\beta+4)(1-k)-\beta+\alpha-4}{\beta-\alpha+4}} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+\frac{(\beta-\alpha+4)(k-1)}{\beta-\alpha+4}} \hat{r}$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{(1-k)-1} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} \hat{r}$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{-k} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} \hat{r}$$

So

$$\nabla \cdot \mathbf{F} = \nabla \cdot (-\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{-k} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} \hat{r})$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} \nabla \cdot (r^{-k}\hat{r})$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} (2k\pi\delta(r))$$

The next section of the appendix will cover specific parameter values that will work.

Coggeshall 9

$$\rho = \rho_0 r^{-\frac{2\beta+k+7}{\alpha}} t^{-\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}$$
$$T = \frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-2\beta-k-7)} \frac{r^2}{t^2}$$

$$\text{Define } T_0 \text{ as } T_0 = \frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-2\beta-k-7)}$$
$$\nabla T = 2T_0 \frac{r}{t^2} \hat{r}$$
$$-\frac{4ac\lambda_0}{3}\rho^\alpha T^{\beta+3}\nabla T = -\frac{4ac\lambda_0}{3}\rho_0^\alpha r^{-(2\beta+k+7)} t^{-\frac{\alpha(k+1)-2\beta-k-7}{2+(\gamma-1)(k+1)}} T_0^{\beta+4} t^{-2(\beta+4)} r^{2(\beta+3)} 2r\hat{r}$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{-k} t^{-2\beta-8} \hat{r}$$
$$\nabla \cdot \mathbf{F} = \nabla \cdot (-\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} r^{-k} t^{-2\beta-8} \hat{r})$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} t^{-2\beta-8} \nabla \cdot (r^{-k}\hat{r})$$
$$= -\frac{4ac\lambda_0}{3}\rho_0^\alpha T_0^{\beta+4} t^{-2\beta-8} (2k\pi\delta(r))$$

D Parameters Where Coggeshall 8 and 9 are Valid at the Origin

The original equation for the conservation of energy with the heat flux after the assumption of radiation energy and momentum being negligible is

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho u^2 + E) + \nabla \cdot [(\frac{1}{2}\rho u^2 + E + P)\mathbf{u} + \mathbf{F}] = 0$$

Without any quantity explicitly containing a delta distribution, the only way to cancel a delta distribution in the $\nabla \cdot \mathbf{F}$ term is for the $\nabla \cdot [(\frac{1}{2}\rho u^2 + E + P)\mathbf{u}]$ to contain an equal and opposite delta distribution.

Coggeshall 8

From the previous section

$$\nabla \cdot \mathbf{F} = -\frac{4ac\lambda_0}{3}\rho_0^{\alpha} T_0^{\beta+4} t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)}(2k\pi\delta(r))$$

Now starting with

$$\begin{aligned}
\nabla \cdot [(\frac{1}{2}\rho u^2 + E + P)\mathbf{u}] &= \nabla \cdot (\frac{1}{2}\rho u^2 \mathbf{u} + \frac{\Gamma}{\gamma - 1}\rho T \mathbf{u} + \Gamma \rho T \mathbf{u}) \\
&= \nabla \cdot (\frac{1}{2}\rho u^2 \mathbf{u} + \frac{\Gamma\gamma}{\gamma - 1}\rho T \mathbf{u})
\end{aligned}$$

$$\begin{aligned}
\rho &= \rho_0 r^{\frac{k-1}{\beta-\alpha+4}} t^{-k-1-\frac{k-1}{\beta-\alpha+4}} \\
\mathbf{u} &= \frac{r}{t}\hat{r} \\
T &= T_0 r^{\frac{1-k}{\beta-\alpha+4}} t^{(1-\gamma)(k+1)+\frac{k-1}{\beta-\alpha+4}}
\end{aligned}$$

$$\begin{aligned}
\nabla \cdot [(\frac{1}{2}\rho u^2 + E + P)\mathbf{u}] &= \nabla \cdot (\frac{1}{2}\rho_0 r^{\frac{k-1}{\beta-\alpha+4}} t^{-k-1-\frac{k-1}{\beta-\alpha+4}} \frac{r^3}{t^3}\hat{r} \\
&\quad + \frac{\Gamma\gamma}{\gamma - 1}\rho_0 r^{\frac{k-1}{\beta-\alpha+4}} t^{-k-1-\frac{k-1}{\beta-\alpha+4}} T_0 r^{\frac{1-k}{\beta-\alpha+4}} t^{(1-\gamma)(k+1)+\frac{k-1}{\beta-\alpha+4}} \frac{r}{t}\hat{r}) \\
&= \nabla \cdot (\frac{\rho_0}{2} r^{3+\frac{k-1}{\beta-\alpha+4}} t^{-k-4-\frac{k-1}{\beta-\alpha+4}}\hat{r} + \frac{\Gamma\gamma}{\gamma-1}\rho_0 T_0 r t^{-1-\gamma(k+1)}\hat{r}) \\
&= \nabla \cdot (\frac{\rho_0}{2} r^{3+\frac{k-1}{\beta-\alpha+4}} t^{-k-4-\frac{k-1}{\beta-\alpha+4}}\hat{r}) + \frac{\Gamma\gamma}{\gamma-1}\rho_0 T_0 t^{-1-\gamma(k+1)}
\end{aligned}$$

So to produce a delta distribution, $r^{3+\frac{k-1}{\beta-\alpha+4}}$ must be equal to r^{-k}

$$\begin{aligned}
-k &= 3 + \frac{k-1}{\beta-\alpha+4} \\
-k-3 &= \frac{k-1}{\beta-\alpha+4} \\
(-k-3)(\beta-\alpha+4) &= k-1 \\
(\beta-\alpha+4) &= \frac{k-1}{-k-3} \\
\beta &= \alpha - \frac{k-1}{k+3} - 4
\end{aligned}$$

Now for the resulting delta distribution to cancel with the one produced by the heat flux, they must have the same time dependence.

$$
\begin{aligned}
t^{(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1)} &= t^{-k-4-\frac{k-1}{\beta-\alpha+4}} \\
(\beta-\alpha+4)(k+1)-\gamma(\beta+4)(k+1)+(k-1) &= -k-4-\frac{k-1}{\beta-\alpha+4} \\
-\frac{k-1}{k+3}(k+1)-\gamma(\alpha-\frac{k-1}{k+3})(k+1)+(k-1) &= -k-4-\frac{k-1}{-\frac{k-1}{k+3}} \\
-\frac{k-1}{k+3}(k+1)-\gamma(\alpha-\frac{k-1}{k+3})(k+1)+(k-1) &= -k-4+k+3 \\
-\frac{k-1}{k+3}(k+1)-\gamma(\alpha-\frac{k-1}{k+3})(k+1) &= -k \\
-\frac{k-1}{k+3}-\gamma(\alpha-\frac{k-1}{k+3}) &= -\frac{k}{k+1} \\
-(k-1)-\gamma(\alpha(k+3)-k+1) &= -\frac{k^2+3k}{k+1} \\
-\gamma(\alpha(k+3)-k+1) &= -\frac{k^2+3k}{k+1}+k-1 \\
\gamma(\alpha(k+3)-k+1) &= \frac{k^2+3k-k^2+1}{k+1} \\
\gamma(\alpha(k+3)-k+1) &= \frac{3k+1}{k+1} \\
\gamma &= \frac{3k+1}{(k+1)(\alpha(k+3)-k+1)}
\end{aligned}
$$

Finally, they must have the same magnitude

$$
\begin{aligned}
\frac{\rho_0}{2} &= \frac{4ac\lambda_0}{3}\rho_0^{\alpha}T_0^{\beta+4} \\
\rho_0^{1-\alpha} &= \frac{8ac\lambda_0}{3}T_0^{\beta+4} \\
\rho_0 &= \left(\frac{8ac\lambda_0}{3}T_0^{\beta+4}\right)^{\frac{1}{1-\alpha}}
\end{aligned}
$$

So, as long as this set of equations is satisfied, Coggeshall solution 8 is a solution.

$$\beta = \alpha - \frac{k-1}{k+3} - 4 \tag{40}$$

$$\gamma = \frac{3k+1}{(k+1)(\alpha(k+3)-k+1)} \tag{41}$$

$$\rho_0 = \left(\frac{8ac\lambda_0}{3}T_0^{\beta+4}\right)^{\frac{1}{1-\alpha}} \tag{42}$$

Coggeshall 9

Repeating the same argument for Coggeshall solution 9

$$\begin{aligned}\nabla\cdot[(\frac{1}{2}\rho u^2+E+P)\mathbf{u}] &= \nabla\cdot(\frac{1}{2}\rho u^2\mathbf{u}+\frac{\Gamma}{\gamma-1}\rho T\mathbf{u}+\Gamma\rho T\mathbf{u})\\ &= \nabla\cdot(\frac{1}{2}\rho u^2\mathbf{u}+\frac{\Gamma\gamma}{\gamma-1}\rho T\mathbf{u})\end{aligned}$$

$$\begin{aligned}\rho &= \rho_0 r^{-\frac{2\beta+k+7}{\alpha}} t^{-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}\\ u &= \frac{2}{2+(\gamma-1)(k+1)}\frac{r}{t}\\ T &= \frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-2\beta-k-7)}\frac{r^2}{t^2} = T_0\frac{r^2}{t^2}\end{aligned}$$

$$\begin{aligned}\nabla\cdot[(\frac{1}{2}\rho u^2+E+P)\mathbf{u}] &= \nabla\cdot(\frac{1}{2}\rho_0 r^{-\frac{2\beta+k+7}{\alpha}} t^{-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}(\frac{2}{2+(\gamma-1)(k+1)})^3\frac{r^3}{t^3}\hat{r}\\ &\quad+\frac{\Gamma\gamma}{\gamma-1}\rho_0 r^{-\frac{2\beta+k+7}{\alpha}} t^{-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}\frac{2}{2+(\gamma-1)(k+1)}\frac{r}{t}T_0\frac{r^2}{t^2}\hat{r})\\ &= \nabla\cdot(\frac{\rho_0}{2}(\frac{2}{2+(\gamma-1)(k+1)})^3 r^{\frac{3\alpha-2\beta-k-7}{\alpha}} t^{-3-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}\hat{r}\\ &\quad+\frac{\Gamma\gamma}{\gamma-1}\frac{2}{2+(\gamma-1)(k+1)}\rho_0 T_0 r^{\frac{3\alpha-2\beta-k-7}{\alpha}} t^{-3-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}\hat{r})\\ &= \nabla\cdot[(\frac{1}{2}(\frac{2}{2+(\gamma-1)(k+1)})^2\\ &\quad+\frac{\Gamma\gamma}{\gamma-1}T_0)\frac{2}{2+(\gamma-1)(k+1)}\rho_0 r^{\frac{3\alpha-2\beta-k-7}{\alpha}} t^{-3-2\frac{\alpha(k+1)-2\beta-k-7}{\alpha[2+(\gamma-1)(k+1)]}}\hat{r}]\end{aligned}$$

For this to produce a delta distribution

$$\begin{aligned}\frac{3\alpha-2\beta-k-7}{\alpha} &= -k\\ 3\alpha-2\beta-k-7 &= -\alpha k\\ (3+k)\alpha &= 2\beta+k+7\\ \alpha &= \frac{2\beta+k+7}{k+3}\end{aligned}$$

Of course, they must have the same time dependence

$$
\begin{aligned}
-\beta - 8 &= -3 - 2\frac{\alpha(k+1) - 2\beta - k - 7}{\alpha[2 + (\gamma - 1)(k+1)]} \\
-\beta - 5 &= -2\frac{\alpha(k+1) - 2\beta - k - 7}{\alpha[2 + (\gamma - 1)(k+1)]} \\
\frac{\beta + 5}{2} &= \frac{\alpha(k+1) - 2\beta - k - 7}{\alpha[2 + (\gamma - 1)(k+1)]} \\
\frac{\beta + 5}{2} &= \frac{\alpha(k+1) - (3+k)\alpha}{\alpha[2 + (\gamma - 1)(k+1)]} \\
\frac{\beta + 5}{2} &= \frac{(k+1) - (3+k)}{2 + (\gamma - 1)(k+1)} \\
(\beta + 5)[2 + (\gamma - 1)(k+1)] &= -4 \\
2 + (\gamma - 1)(k+1) &= -\frac{4}{\beta + 5} \\
(\gamma - 1)(k+1) &= -2 - \frac{4}{\beta + 5} \\
\gamma - 1 &= -\frac{2}{k+1} - \frac{4}{(\beta + 5)(k+1)} \\
\gamma &= \frac{k-1}{k+1} - \frac{4}{(\beta + 5)(k+1)}
\end{aligned}
$$

To make the coefficients be equal in magnitude so that the delta distributions cancel

$$\begin{aligned}
\frac{4ac\lambda_0}{3}\rho_0^{\alpha}T_0^{\beta+4} &= \left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{2}{2+(\gamma-1)T_0)(k+1)}\rho_0 \\
\rho_0^{\alpha-1} &= \left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}T_0^{-(\beta+4)} \\
\rho_0 &= \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}T_0^{-(\beta+4)}\right]^{\frac{1}{\alpha-1}} \\
&= \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}\right]^{\frac{1}{\alpha-1}} \\
&\quad\cdot\left[\frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-2\beta-k-7)}\right]^{\frac{-(\beta+4)}{\alpha-1}} \\
&= \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}\right]^{\frac{1}{\alpha-1}} \\
&\quad\cdot\left[\frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-\alpha(k+3))}\right]^{\frac{-(\beta+4)}{\alpha-1}} \\
&= \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}\right]^{\frac{1}{\alpha-1}} \\
&\quad\cdot\left[\frac{2\alpha(\gamma-1)(k+1)}{\Gamma[2+(\gamma-1)(k+1)]^2(-\alpha(k+1))}\right]^{\frac{-(\beta+4)}{\alpha-1}} \\
&= \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}\right]^{\frac{1}{\alpha-1}} \\
&\quad\cdot\left[\frac{2(\gamma-1)}{\Gamma[2+(\gamma-1)(k+1)]^2}\right]^{\frac{-(\beta+4)}{\alpha-1}}
\end{aligned}$$

So to summarize, Coggeshall solution 9 is a solution if the following system of equations is satisfied.

$$\alpha = \frac{2\beta+k+7}{k+3} \tag{43}$$

$$\gamma = \frac{k-1}{k+1} - \frac{4}{(\beta+5)(k+1)} \tag{44}$$

$$\rho_0 = \left[\left(\frac{1}{2}\left(\frac{2}{2+(\gamma-1)(k+1)}\right)^2+\frac{\Gamma\gamma}{\gamma-1}\right)\frac{3}{4ac\lambda_0[2+(\gamma-1)T_0)(k+1)]}\right]^{\frac{1}{\alpha-1}} \tag{45}$$

$$\cdot\left[\frac{2(\gamma-1)}{\Gamma[2+(\gamma-1)(k+1)]^2}\right]^{\frac{-(\beta+4)}{\alpha-1}} \tag{46}$$

E Coggeshall Solution 9 with Different Values of β

$\beta = -3.10$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{1.87}t^{0.57}} \\
T(r,t) =& \frac{2.81r^2}{t^2} \\
F(r,t) =& \frac{-3381.60}{r^2t^{2.65}} \\
P(r,t) =& \frac{1.88r^{0.13}}{t^{2.57}} \\
S(r,t) =& \frac{1.77r}{t}
\end{aligned}
$$

$\beta = -3.60$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{1.20}t^{0.90}} \\
T(r,t) =& \frac{0.47r^2}{t^2} \\
F(r,t) =& \frac{-984.73}{r^2t^{2.15}} \\
P(r,t) =& \frac{0.31r^{0.80}}{t^{2.90}} \\
S(r,t) =& \frac{0.72r}{t}
\end{aligned}
$$

$\beta = -3.75$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{rt} \\
T(r,t) =& \frac{0.38r^2}{t^2} \\
F(r,t) =& \frac{-1043.9}{r^2t^2} \\
P(r,t) =& \frac{0.25r}{t^3} \\
S(r,t) =& \frac{0.65r}{t}
\end{aligned}
$$

$\beta = -3.80$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.93}t^{1.03}} \\
T(r,t) =& \frac{0.35r^2}{t^2} \\
F(r,t) =& \frac{-1081.78}{r^2t^{1.95}} \\
P(r,t) =& \frac{0.23r^{1.07}}{t^{3.03}} \\
S(r,t) =& \frac{0.62r}{t}
\end{aligned}
$$

$\beta = -3.90$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.80}t^{1.10}} \\
T(r,t) =& \frac{0.31r^2}{t^2} \\
F(r,t) =& \frac{-1186.93}{r^2t^{1.85}} \\
P(r,t) =& \frac{0.21r^{1.20}}{t^{3.10}} \\
S(r,t) =& \frac{0.59r}{t}
\end{aligned}
$$

$\beta = -4.00$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.67}t^{1.17}} \\
T(r,t) =& \frac{0.28r^2}{t^2} \\
F(r,t) =& \frac{-1333.33}{r^2t^{1.75}} \\
P(r,t) =& \frac{0.19r^{1.33}}{t^{-3.17}} \\
S(r,t) =& \frac{0.56r}{t}
\end{aligned}
$$

$\beta = -4.10$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.53}t^{1.23}} \\
T(r,t) =& \frac{0.26r^2}{t^2} \\
F(r,t) =& \frac{-1528.16}{r^2t^{1.65}} \\
P(r,t) =& \frac{0.17r^{1.47}}{t^{3.23}} \\
S(r,t) =& \frac{0.53r}{t}
\end{aligned}
$$

$\beta = -4.20$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.40}t^{1.30}} \\
T(r,t) =& \frac{0.23r^2}{t^2} \\
F(r,t) =& \frac{-1782.20}{r^2t^{1.55}} \\
P(r,t) =& \frac{0.16r^{1.60}}{t^{3.30}} \\
S(r,t) =& \frac{0.51r}{t}
\end{aligned}
$$

$\beta = -4.40$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{r^{0.13}t^{1.43}} \\
T(r,t) =& \frac{0.20r^2}{t^2} \\
F(r,t) =& \frac{-2533.69}{r^2t^{1.35}} \\
P(r,t) =& \frac{0.13r^{1.87}}{t^{3.43}} \\
S(r,t) =& \frac{0.47r}{t}
\end{aligned}
$$

$\beta = -4.50$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{1}{t^{1.50}} \\
T(r,t) =& \frac{0.19r^2}{t^2} \\
F(r,t) =& \frac{-3079.20}{r^2 t^{1.25}} \\
P(r,t) =& \frac{0.12r^2}{t^{-3.50}} \\
S(r,t) =& \frac{0.46r}{t}
\end{aligned}
$$

$\beta = -5.25$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{r}{t^2} \\
T(r,t) =& \frac{0.12r^2}{t^2} \\
F(r,t) =& \frac{-17939.12}{r^2 t^{0.50}} \\
P(r,t) =& \frac{0.08r^3}{t^4} \\
S(r,t) =& \frac{0.37r}{t}
\end{aligned}
$$

$\beta = -6.00$:

$$
\begin{aligned}
u(r,t) =& \frac{r}{2t} \\
\rho(r,t) =& \frac{r^2}{t^{2.50}} \\
T(r,t) =& \frac{0.09r^2}{t^2} \\
F(r,t) =& \frac{-151703.70}{r^2 t^{0.25}} \\
P(r,t) =& \frac{0.06r^4}{t^{4.50}} \\
S(r,t) =& \frac{0.32r}{t}
\end{aligned}
$$

Using High-Fidelity Radiation Transport Methods to Supplement the Diffusion Approximation at Material Interfaces

(Todd Urbatsch and Scott Runnels, mentors)

LA-UR-13-26532
Approved for public release; distribution is unlimited.

Title: Using High-Fidelity Radiation Transport Methods to Supplement the Diffusion Approximation at Material Interfaces

Author(s): Ruiz, Daniel E
Stephey, Laurie A

Intended for: Show to colleages at home institutions
Report

Issued: 2013-08-19

Using High-Fidelity Radiation Transport Methods to Supplement the Diffusion Approximation at Material Interfaces

Daniel E. Ruiz and Laurie A. Stephey
Mentors: Todd Urbatsch and Scott Runnels

16 August 2013

Acknowledgements

Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the Nuclear Security Administration of the U. S. Department of Energy under contract DE-AC52-06NA25396.

1 Abstract

Today, many radiation-hydrodynamics simulations use methods that combine hydrodynamics and radiation diffusion algorithms. The diffusion model has been used because it provides faster calculations for the radiation scalar flux than the full radiation transport model. [5, 7, 8] However, it is well-known that diffusion cannot resolve *"transport effects"* at material interfaces. This could cause erroneous predictions when coupling the radiation results with a material hydrodynamics calculation. In this work, the *"transport effects"* are thoroughly characterized using IMC and S_N transport methods. Based on the information obtained from this study, new models based on diffusion that can capture these effects are developed. These models are subsequently compared with the full radiation transport model. It is hoped that the new models can be used to provide improved predictions in rad-hydro simulations, such as in ICF capsule implosions.

2 Keywords

Thermal Radiation Transport, Radiation Transport Boundary Effects, S_N Approximation, Implicit Monte Carlo, Hybrid Transport-Diffusion Method, Radiation Hydrodynamics.

Contents

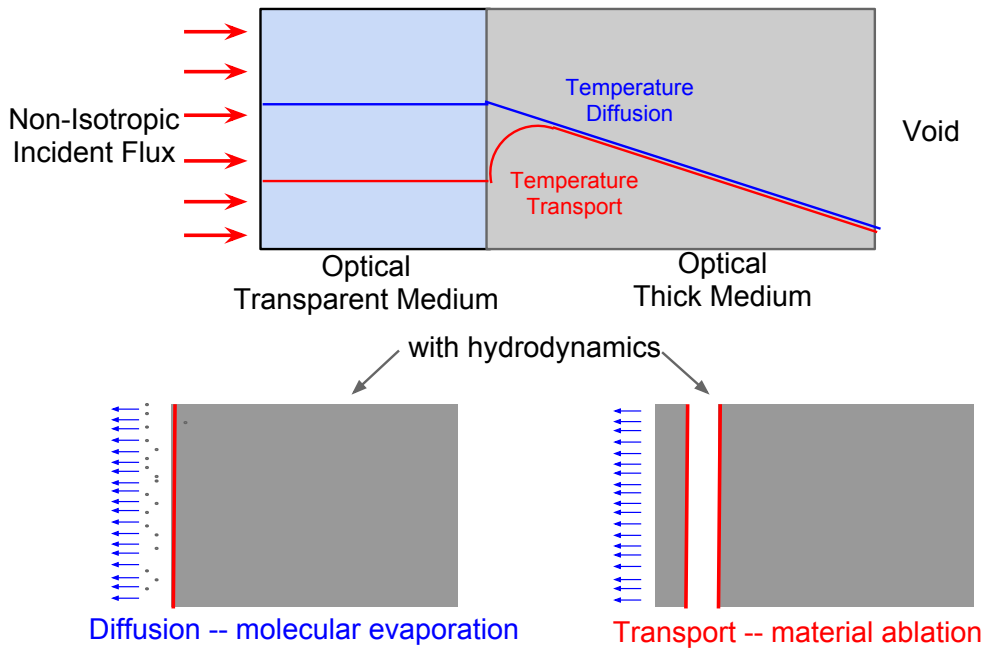


Figure 1: Different hydrodynamic effects might occur when modeling radiation with diffusion and transport methods.

3 Introduction

Thermal radiation transport coupled with material hydrodynamics is an extremely complex engineering problem. AT LANL, the FLAG code combines an ALE method to model the material hydrodynamics and a radiation diffusion method to simulate the radiation field. The radiation diffusion model is a commonly used transport calculation method because it is relatively inexpensive and can provide good results in asymptotic limits. [5, 7, 8] However, as a result of the integration inherent in the method, it cannot provide a detailed description of radiation transport in many circumstances.

It is well-known that radiation diffusion cannot capture *"transport effects"* at material boundaries and/or interfaces. As an example, consider a situation where a normally incident photon flux travels through an optically transparent medium (air) into an opaque medium (plastic). The diffusion method predicts the highest temperature for the thick medium at the material interface. On the other hand, a full radiation transport model would predict maximum temperatures at approximately 1 mfp inside the thick material. When these results are coupled in a hydro simulation, the diffusion-hydro coupling could predict molecular evaporation of the thick material surface. In contrast, the transport-hydro coupling could predict melting and less material ablation. These effects are illustrated in Figure 1. It is therefore imperative to correctly model the radiation at material interfaces in order to simulate the hydrodynamic effects accurately.

The goal of this work is to study the radiation *"transport effects"* at optically thin-thick material interfaces without considering material hydrodynamics. As a first objective, several radiation transport methods will be used in order to carefully characterize the material interface for a wide variety of problem parameters, such as: opacity, density, grid refinement, incoming photon flux, etc. The second objective of this work is to study new models based on diffusion that can capture the *"transport effects"*. It is

hoped that these new models will be implemented into the LANL FLAG code in order to achieve better predictions.

The work is organized as follows: in Section 4, the basic thermal transport mathematical model is presented. The main assumptions and limitations of the model are discussed. Section 5 presents the precise mathematical statement of the problem studied. Then in Section 6, the main benchmark methods used for modeling thermal radiation transport are discussed. The primary methods used were the Implicit Monte Carlo (IMC) method, the deterministic S_N method and the Radiation Diffusion method. Next, a thorough characterization of the boundary interface is presented in Section 7. After a thorough understanding of the physical problem is obtained, new models based on diffusion are developed and presented in Section 8. The results of these models are compared with the benchmark models in Section 9. Conclusions are given in Section 11.

4 Thermal Radiation Transfer: Basic Equations and Approximations

The equations of thermal radiation transfer are a mathematical model that describes the physical processes of photon scattering, absorption, and emission by a high-energy background material. In this work, we will investigate the properties of the 1-D grey equations (as opposed to the multi-group equations). The governing equations are the following: [5, 15]

$$\frac{1}{c}\frac{\partial}{\partial t}\psi(x,\mu,t) + \mu\frac{\partial}{\partial x}\psi(x,\mu,t) + \Sigma_a(x,T)\psi(x,\mu,t) = \frac{1}{2}\Sigma_a(x,T)acT^4(x,t) \tag{1}$$

$$\frac{\partial}{\partial t}e(x,t) = \Sigma_a(x,T)\int_{-1}^{1}\psi(x,\mu',t)\,d\mu' \; - \Sigma_a(x,T)acT^4(x,t) \tag{2}$$

where ψ is the specific intensity, $\mu = \cos(\theta) \in [-1,1]$, θ is the angle of the of the photon packet direction with respect to the x-axis, $\Sigma(x,T)$ is the photon macroscopic absorption cross-section, $a = 7.5657(10)^{-16}Jm^{-3}K^{-4}$ radiation constant, c is the speed of light, T is the material temperature and e is the material internal energy density. The common equation of state for the material internal energy density is given by:

$$\frac{\partial}{\partial t}e(x,t) = C_v(x,T)\frac{\partial}{\partial t}T(x,t) \tag{3}$$

where $C_v(x,T)$ is the specific heat capacity of the material. More precision on the initial and boundary conditions will be described later when the problem statement is given.

Before we describe the problem in detail, we will first describe the assumptions implicit in the governing equations above(1-3): [18]

- The photon density is large, so that fluctuations caused by individual photon dynamics can be ignored. This simplification is key in order to derive the Boltzmann transport equation (1).
- Interference effects are ignored since the transport equation is an equation for intensities rather than amplitudes. Photons are necessarily treated as incoherent.
- No diffraction or reflection is possible. These phenomena depend upon interference among the waves arising from different scattering centers that scatter the same photon.
- Light polarization is neglected.

- Refraction and dispersion are neglected. That is, the refractive index n is taken as unity. Photons always travel at the speed of light, regardless of the material in which they are traveling.
- The material medium is assumed to be isotropic. Consequently, the photon emission is also isotropic.
- Assuming that the material is in local thermodynamic equilibrium, its energy is described solely by specifying its temperature. The general equations of heat transfer contain terms that describe convective, conductive, and radiative source and loss terms. However, here we assume that the radiative term is sufficient to describe the heat transfer. Since the material is in local thermodynamic material, its radiation emission follows the Stephan-Boltzmann radiation law.
- A 1D geometry is used. Hence, we assume that there is an azimuthal symmetry for the problem.
- Grey absorption cross-sections are assumed. This allows a single energy group description of the Boltzmann transport equation to be used.

5 Problem Statement

In this section, we will describe the primary problem of interest. Many simplifications have been made to make this problem tractable. In future work, some of these assumptions may be relaxed so the problem can be generalized. The statement of the problem is as follows:

Statement of problem: Calculate the values of $\psi(x,\mu,t)$ and $T(x,t)$ given the following equations describing the thermal radiation transport:

$$\frac{1}{c}\frac{\partial}{\partial t}\psi(x,\mu,t)+\mu\frac{\partial}{\partial x}\psi(x,\mu,t)+\Sigma_a(x,T)\psi(x,\mu,t)=\frac{1}{2}\Sigma_a(x,T)acT^4(x,t) \tag{4}$$

$$C_v(x,T)\frac{\partial}{\partial t}T(x,t)=\Sigma_a(x,T)\int_{-1}^{1}\psi(x,\mu',t)\,d\mu' \ -\Sigma_a(x,T)acT^4(x,t) \tag{5}$$

with the initial conditions

$$\psi(x,\mu,t=0)=0,\forall x\in\mathbb{R}_+,\forall\mu\in[-1,1] \tag{6}$$

$$T(x,t=0)=0,x\in\mathbb{R}_+ \tag{7}$$

and the boundary conditions

$$\psi(x=0,\mu,t)=\psi_l(\mu),\forall\mu\in[0,1],\forall t\in\mathbb{R}_+ \tag{8}$$

$$\psi(x=+\infty,\mu,t)=0,\forall\mu\in[-1,1],\forall t\in\mathbb{R}_+ \tag{9}$$

where the material properties are given by

$$\Sigma_a(x,T)=\begin{cases}\Sigma_{a1} & 0\le x<L_1\\ \Sigma_{a2} & x\ge L_1\end{cases} \tag{10}$$

$$C_v(x,T)=\begin{cases}C_{v1} & 0\le x<L_1\\ C_{v2} & x\ge L_1\end{cases} \tag{11}$$

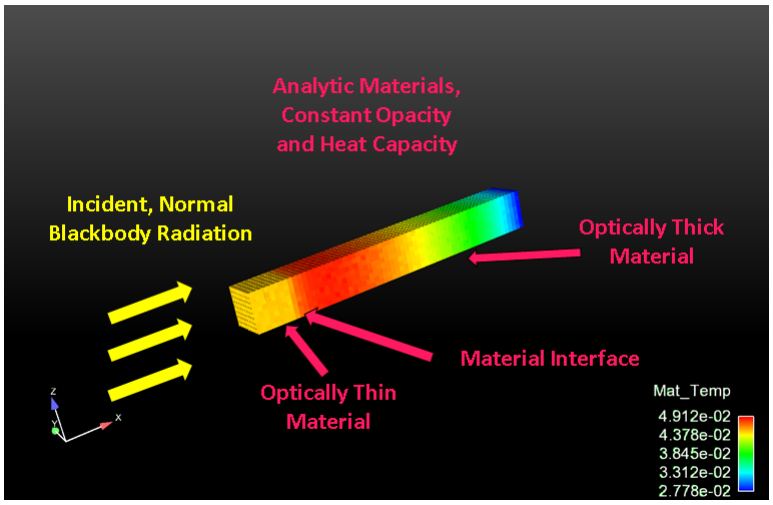


Figure 2: Problem Setup with Temperature distribution from IMC results.

where Σ_{a1}, Σ_{a2}, C_{v1} *and* C_{v2} *are constants.*

As a summary, we wish to describe the thermal radiative transfer for the time-dependent half-space problem: $(x,t) \in \mathbb{R}_+ \times \mathbb{R}_+$. The initial conditions for the specific intensity and the temperature field are zero. At the left-side of the half-space domain, an entering photon flux ψ_l is given. At $x = +\infty$, the specific intensity is zero. The computational grid is divided into two regions composed of two materials. The length of the first material is L_1. We assume that the material properties (Σ, C_v) remain constant for each material. This physical problem is summarized in Figure 2. The main goal of this work will be to study the radiative transfer boundary layer effects at the material interface.

Remark: An important note is that the system of equations (4-11) contain an energy conservation structure. [1] By integrating equation (4) with respect to μ and labeling the scalar flux $\phi(x,t) = \int_0^1 \psi(x,\mu',t)\,d\mu'$ and the current flux $J(x,t) = \int_{-1}^1 \psi(x,\mu',t)\mu'\,d\mu'$, we obtain:

$$\frac{1}{c}\frac{\partial}{\partial t}\phi(x,t) + \frac{\partial}{\partial x}J(x,t) + \Sigma_a(x,T)\phi(x,t) = acT^4(x,t)$$

We add the last equation to equation 5 and we integrate over the entire half-space. The obtained conservation law is:

$$\int_0^{+\infty} \frac{\partial}{\partial t}\left\{\frac{1}{c}\phi(x',t) + \rho(x')c_v(x')T(x',t)\right\} dx' = \int_0^1 \psi_l(\mu')\mu'\,d\mu' = J_l \tag{12}$$

The last equation shows that the change of the material internal energy and the radiation field energy equals the incoming photon current flux.

Remark: In steady state, the material temperature field will be in equilibrium with the radiation scalar flux field. In this situation, the specific intensity is described by the stationary pure transport equation which was studied by Case, Davidson and Chandrasekhar: [4]

$$\mu\frac{\partial}{\partial x}\psi(x,\mu,t) + \Sigma_a(x)\psi(x,\mu,t) = \frac{1}{2}\Sigma_a(x)\int_{-1}^1 \psi(x,\mu',t)\,d\mu'$$

Remark: In order to adimensionalize the governing equations (4-11), we introduce the following dimensionless parameters:

$$\bar{t} = c\Sigma_{a2}t \quad \bar{x} = \Sigma_{a2}x \quad \bar{\psi} = \psi/(\sigma T_{in}^4) \quad \bar{T} = T/T_{in} \tag{13}$$

where σ is the Stephan-Boltzmann coefficient and $\sigma T_{in}^4 = \int_{-1}^{1} \psi_l(\mu')\mu'\, d\mu'$. When introducing the variable changes into equations (4-5), the equations become:

$$\frac{\partial}{\partial \bar{t}}\bar{\psi}(\bar{x},\mu,\bar{t}) + \mu\frac{\partial}{\partial \bar{x}}\bar{\psi}(\bar{x},\mu,\bar{t}) + \frac{\Sigma_a(x)}{\Sigma_{a2}}\bar{\psi}(\bar{x},\mu,\bar{t}) = 2\frac{\Sigma_a(x)}{\Sigma_{a2}}\bar{T}^4(\bar{x},\bar{t}) \tag{14}$$

$$\frac{\partial}{\partial \bar{t}}\bar{T}(\bar{x},\bar{t}) = \frac{1}{4}\frac{\Sigma_a(x)}{\Sigma_{a2}}\frac{\sigma T_{in}^3}{\rho(\bar{x})c_v(\bar{x})}\left\{\int_{-1}^{1}\bar{\psi}(\bar{x},\mu',\bar{t})\,d\mu' - 4\bar{T}^4(\bar{x},\bar{t})\right\} \tag{15}$$

As shown in the previous equation, the variable change introduces a dimensionless parameter: $B_0 = \dfrac{\sigma T_i n^3}{\rho(\bar{x})c_v(\bar{x})}$, which is called the Boltzmann number. This dimensionless parameter describes how fast the material temperature will change with the surrounding radiation field. In the case in which $B_0 \ll 1$, the material internal energy remains almost stationary and does not change. If the initial temperature is zero, the material would never re-emit the captured photons. Hence, the source term of equation (14) is null and the resulting transport equation with constant absorption would be easily solvable. On the other hand, when $B_0 \gg 1$, the material temperature immediately achieves equilibrium with the surrounding radiation field. In this case, the transport equation for the radiation field can be very well approximated by :

$$\frac{\partial}{\partial \bar{t}}\bar{\psi}(\bar{x},\mu,\bar{t}) + \mu\frac{\partial}{\partial \bar{x}}\bar{\psi}(\bar{x},\mu,\bar{t}) + \frac{\Sigma_a(x)}{\Sigma_{a2}}\bar{\psi}(\bar{x},\mu,\bar{t}) = \frac{1}{2}\frac{\Sigma_a(x)}{\Sigma_{a2}}\int_{-1}^{1}\bar{\psi}(\bar{x},\mu',\bar{t})\,d\mu'$$

which is the time-dependent Case problem with isotropic scattering.

6 Radiation Transport Methods

In this section, the three principal benchmark methods used for our studies are described. These are the *Implicit Monte Carlo Method* (IMC), the *Discrete Ordinates (S_N) Method* and the *Radiation Diffusion Method*. For IMC, the algorithm used was the previously developed LANL MILAGRO code. [23] For the S_N method, the algorithm used was based on a numerical diamond scheme. The S_N and Diffusion algorithms were developed *ad-hoc* in order to provide more flexibility. These codes were compared with the deterministic LANL SERRANO algorithm, on which the S_N method used in this work is based. [3]

6.1 Implicit Monte Carlo (IMC) Method

Monte Carlo methods, in general, can only be used to solve linear equations. Due to the high degree of non-linearity inherent in the radiative transfer equations, traditional Monte Carlo methods cannot be used to solve this system. A method proposed by Fleck and Cummings [10] allows Monte Carlo calculations to be used to solve a linearized version of the set of radiative transfer equations time-implicitly.

The following overview draws on work published by McClarren and Urbatsch [15]. As above, we begin with the grey thermal radiation transfer equations with no scattering:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\Omega}\cdot\nabla I + \sigma I = \frac{1}{4\pi}\sigma acT^4 \tag{16}$$

$$\frac{\partial u_m}{\partial t} = \sigma \left(\int_{4\pi} I d\hat{\Omega} - acT^4 \right) + S \tag{17}$$

To linearize the radiative transfer equations, Fleck and Cummings define an energy density variable, u_r. This is the value of the radiation energy density when the material and radiation are in equilibrium. [15].

$$u_r = aT^4 \tag{18}$$

It is also common to rewrite the following term as:

$$\frac{\partial u_m}{\partial u_r} = \beta^{-1} \tag{19}$$

Using these definitions, we can now rewrite the radiative transfer equations in the following form:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\Omega} \cdot \nabla I + \sigma I = \frac{1}{4\pi} c\sigma u_r \tag{20}$$

$$\frac{\partial u_r}{\partial t} = \beta\sigma \left(\int_{4\pi} I d\hat{\Omega} - cu_r \right) + \beta S \tag{21}$$

For this scheme to be implicit, an implicit definition of u_r is required. The Fleck and Cummings [10] approach averages over a time step:

$$\frac{{u_r}^{n+1} - {u_r}^n}{\Delta t} = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} dt [\beta\sigma \left(\int_{4\pi} I d\hat{\Omega} - cu_r \right) + \beta S] \tag{22}$$

We can thus determine the average value of u_r:

$$\overline{u_r} \approx \alpha {u_r}^{n+1} + (1-\alpha) {u_r}^n \tag{23}$$

We can substitute this value into the previous equation and perform a time-averaging procedure. After this is completed, it is convenient to define the following factor (commonly called the Fleck factor):

$$f = \frac{1}{1 + \alpha\beta\sigma c\Delta t} \tag{24}$$

which appears in the resulting expression:

$$\overline{u_r} = f {u_r}^n + \frac{1-f}{c} \left(\int_{4\pi} I d\hat{\Omega} + \frac{1}{\sigma} S \right) \tag{25}$$

We can then substitute this expression for $\overline{u_r}$ back into the first radiative transfer equation to find:

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\Omega} \cdot \nabla I + \sigma I = \frac{1}{4\pi}(1-f)\sigma \int_{4\pi} I\hat{\Omega} + \frac{1}{4\pi}(c\sigma f u_r + (1-f)S) \tag{26}$$

which is the now linearized equation that can be solved using Monte-Carlo methods. It should be noted that in general this method is first-order accurate, but if $\alpha = 0.5$ and β and σ are constant, this method can be second-order accurate.

6.2 Discrete Ordinates (S_N)

In a discrete ordinate method, equations (4-5) are solved for each angle μ_k, $k = 1, ..., K$ in a quadrature set with associated weights $w_k{}_{k=1}^{K}$. For this work, a Gauss-Lengendre quadrature was used.

For practical reasons, the computational spatial domain cannot be infinite. We must therefore substitute the spatial half-space by a finite space. We divide the spatial domain into J cells. The position of each cell will be denoted by x_j and will be numbered from $j = 1, ..., J$. A constant time-step is used to discretize the time variable: $t_n = n\Delta t$. By using $\psi_j^{n,k}$, T_j^n as approximations of the solutions $\psi(x_j, \mu_k, t_n)$, $T(x_j, t_n)$, the diamond scheme is the following:[1]

$$\frac{\psi_j^{n+1,k} - \psi_j^{n,k}}{\Delta t} + \mu_k \frac{\psi_{j+1/2}^{n+1/2,k} - \psi_{j-1/2}^{n+1/2,k}}{\Delta x} + \Sigma_{a,j}^{n+1/2} \psi_j^{n+1/2,k} = \frac{1}{2} \Sigma_{a,j}^{n+1/2} ac (T_j^{n+1/2})^4 \tag{27}$$

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \Sigma_{a,j}^{n+1/2} \frac{1}{C_{v,j}^{n+1/2}} \left\{ \sum_{k=1}^{K} \psi_j^{n+1/2,k} w_k - ac (\bar{T}_j^{n+1/2})^4 \right\} \tag{28}$$

with the following diamond relations:

$$\psi_j^{n+1,k} + \psi_j^{n,k} = \psi_{j-1/2}^{n+1/2,k} + \psi_{j+1/2}^{n+1/2,k} \tag{29}$$

$$\psi_j^{n+1/2,k} = \frac{1}{2} \left\{ \psi_{j-1/2}^{n+1/2,k} + \psi_{j+1/2}^{n+1/2,k} \right\} \tag{30}$$

$$T_j^{n+1/2,k} = \frac{1}{2} \left\{ T_j^{n+1} + T_j^n \right\} \tag{31}$$

As always, we use:

$$\Sigma_{a,j}^{n+1/2} = \Sigma_a(x_j, T(x, t_{n+1/2})) \quad C_{v,j}^{n+1/2} = C_v(x_j, T(x_j, t_{n+1/2})) \tag{32}$$

It can be seen that the first diamond equation relates the values of the radiation field at the cell j of the previous and next time steps with the radiation fields at the cell edges at half-time step. From the first diamond relation, we can eliminate the unknown $\psi_j^{n+1,k}$. From the second relation, we can eliminate $\psi_j^{n+1/2,k}$. Finally, with the third relation we can eliminate $T_j^{n+1/2}$. As an example, when substituting these relations into equation (27), we obtain an implicit scheme for the variables $\psi_{j+1/2}^{n+1/2,k}$, $\psi_{j-1/2}^{n+1/2,k}$, and $T_j^{n+1/2,k}$.

The diamond scheme has the advantage of being able to naturally include the boundary conditions for the radiation field. The left boundary condition:

$$\psi_1^{n+1,k} + \psi_1^{n,k} = \psi_l(\mu_k) + \psi_{3/2}^{n+1/2,k}, \forall \mu_k \in [0, 1], \forall n > 0 \tag{33}$$

where $\psi_l(\mu_k)$ is the left incoming flux. As mentioned above, the computational domain cannot be infinite. Instead of the boundary condition (9), we use a vacuum boundary condition at some length L:

$$\psi_J^{n+1,k} + \psi_J^{n,k} = \psi_{J-1}^{n+1/2,k}, \forall \mu_k \in [-1, 0], \forall n > 0 \tag{34}$$

[1] G. Allaire used the diamond scheme to solve the time-dependent neutron transport equation with scattering. Here we modified the scheme in order to model thermal radiation transport. [1]

For the linear transport case, the diamond scheme described above has been proven to be consistent and stable.[2] It has a truncation error of order $\mathcal{O}(\Delta x^2 + \Delta t^2)$. It is important to also note that the present scheme has a fast convergence speed. When substituting the diamond relations into the discretized equations, for $\mu_k > 0$, $\psi_{j+1/2}^{n+1/2,k}$ depends only on $\psi_{j-1/2}^{n+1/2,k}$. Hence, $\psi_{j+1/2}^{n+1/2,k}$ can be calculated quickly from left to right. Once the new values of $\psi_{j+1/2}^{n+1/2,k}$ have been updated, the new temperature field is obtained. The process is iterated until both the radiation and temperature fields satisfy equations (27-28) at the time step n. Since the radiation field variables are calculated almost directly, the only iterated variable at each time step is the temperature field. Hence, there is a high convergence rate for this method.

6.3 Radiation Diffusion Method

The *Radiation Diffusion Method* is commonly used for rad-hydro applications since it is a convenient, fast method to obtain the radiation scalar field. [5,7,8] There are many procedures to obtain the diffusion approximation from the radiation transport equation. One of them involves making the assumption that the radiation specific intensity can be written as follows:

$$\psi(x,\mu,t) = \frac{1}{2}\phi(x,t) + \frac{3}{2}\mu J(x,t) + \mathcal{O}(\mu^2)$$

When inserting the last approximation into the radiation transport equation and taking the zeroth and first moments on μ, the resulting radiation diffusion equations are:

Radiation Diffusion

$$\frac{1}{c}\frac{\partial}{\partial t}\phi - \frac{\partial}{\partial x}\frac{1}{3\Sigma}\frac{\partial}{\partial x}\phi + \Sigma_a\phi = \Sigma_a acT^4 \tag{35}$$

Material Internal Energy

$$C_v(x,T)\frac{\partial}{\partial t}T = \Sigma_a\phi - \Sigma_a acT^4 \tag{36}$$

These equations are much easier to solve than the full transport formulation since the angular dependence on the radiation field no longer needs to be calculated. The main variables to be calculated are only the scalar flux $\phi(x,t)$ and the temperature field T.

To complete the formulation of the diffusion problem, an approximate expression for the scalar flux at the boundary must be obtained. Instead of using the usual Marshak boundary conditions [11, 21], the asymptotic diffusion-limit boundary conditions will be used. [12] The boundary conditions for the scalar flux are written as:

$$2\int_0^1 W(\mu')\psi_l(\mu')\,d\mu' \;= \phi(0,t) - \frac{\lambda}{\Sigma_a(0,T)}\frac{\partial}{\partial x}\phi(0,t), \forall t > 0 \tag{37}$$

$$0 = \phi(L,t) + \frac{\lambda}{\Sigma_a(L,T)}\frac{\partial}{\partial x}\phi(L,t), \forall t > 0 \tag{38}$$

where $\lambda \simeq 0.7104$ is an extrapolation factor and $W(\mu)$ is a weighting function given by $W(\mu) \simeq \mu + \frac{3}{2}\mu^2$.

[2] This is proven for the linear case in neutron transport theory. [1]

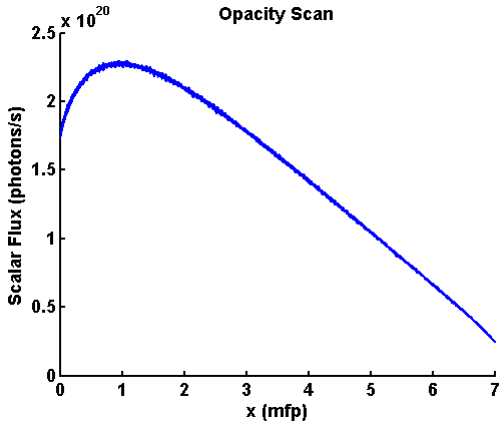


Figure 3: The results of the opacity scan. All 25 scans lie almost exactly on top of each other. The shape of the scalar flux is invariant in terms of mean free path.

7 Large scale parameter study

7.1 Parameter Scans

One approach to solving this problem is to amass a great deal of data and use it to tease out the important physics in this problem.

Several large scale parameter scans were performed for the purpose of isolating the effect of changing a single factor on the scalar photon flux. In each scan, unless otherwise noted, all parameters except the ones discussed were held constant.

It should be noted that in steady-state, once the scalar flux is known, the material temperature can determined in the following manner:

$$T_{mat} = \sqrt[4]{\frac{\phi_s}{ac}} \tag{39}$$

where a is the Stefan-Boltzmann constant.

Once determined, the scalar flux and the material temperature can provide important information for both the radiation and the hydrodynamics aspects of the problem.

7.2 Opacity

The opacity of a material is the approximate "transparency" of the material to photons. In an analytic material, the opacity can be a constant or a function. A scan was performed in which the opacity values of the thin material and the thick material were varied. (It should be noted that the opacity of the thin material was almost negligible, making the system effectively a single thick block in a void.)

The results of this scan are displayed in Figure 3. When the scalar flux and material temperature are plotted as a function of distance, it is apparent that the curves seem to "stretch" with the computational domain. When plotted as a function of mean free path, all 25 curves lie almost exactly on top of each other, an expected result. The shape of the profiles is invariant in terms of the mean free path of the thick material.

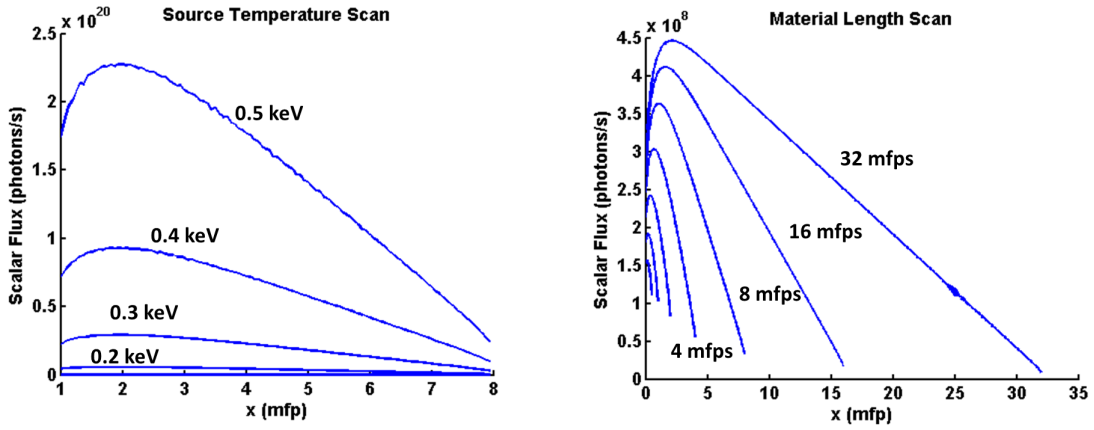


Figure 4: The results of the source temperature scan (left) and the results of the material length scan (right). As the source temperature is increased, the scalar flux profile changes shape and increases in temperature. It can also be seen that as the material becomes longer, the shape of the profile "stretches" with the material and changes in shape.

7.3 Source Temperature

Another scan of incoming flux Planckian source temperature was performed. Photon source temperatures were varied from 0.1 to 0.5 keV. (Smaller temperatures were simulated but the data were not used since it took much longer for the system to relax to steady-state.) As shown in Figure 4, it can be seen that by increasing the source temperature, the scalar flux also increases in a predictable manner.

7.4 Material Length

Another scan of varying lengths of the material was performed. The length of the optically thick material was varied between 1 and 32 mean free paths. As shown in Figure 4, the shape of both profiles of interest again changed in a predictable manner as a function of material length.

Though the opacity of the thick material provides an effectively dimensionless parameter by which the shape of the profiles of interest can be determined, both the source temperature and the material length can not be boiled down to a dimensionless number in this same fashion. Moreover, the profiles are both simultaneously a function of source temperature and material length. This required an additional parameter scan in which both the source temperature and the material length were varied simultaneously. These results can be used to create surfaces of values; this will be discussed in detail in what follows.

8 Methods for capturing *"transport effects"*

8.1 Surrogate Source Term

The goal of performing these large, computationally intensive scans was to develop an understanding of the relevant physical variables in the system. Once we had determined the important parameters in the system, our aim was to independently generate the shapes of the scalar flux profile in a variety of situations using surrogate functions. These surrogate functions, given the opacity of the thick material (and hence the mean free path), the source temperature, and the material length, could be used to generate the appropriate profiles from a lookup table without performing a full transport calculation.

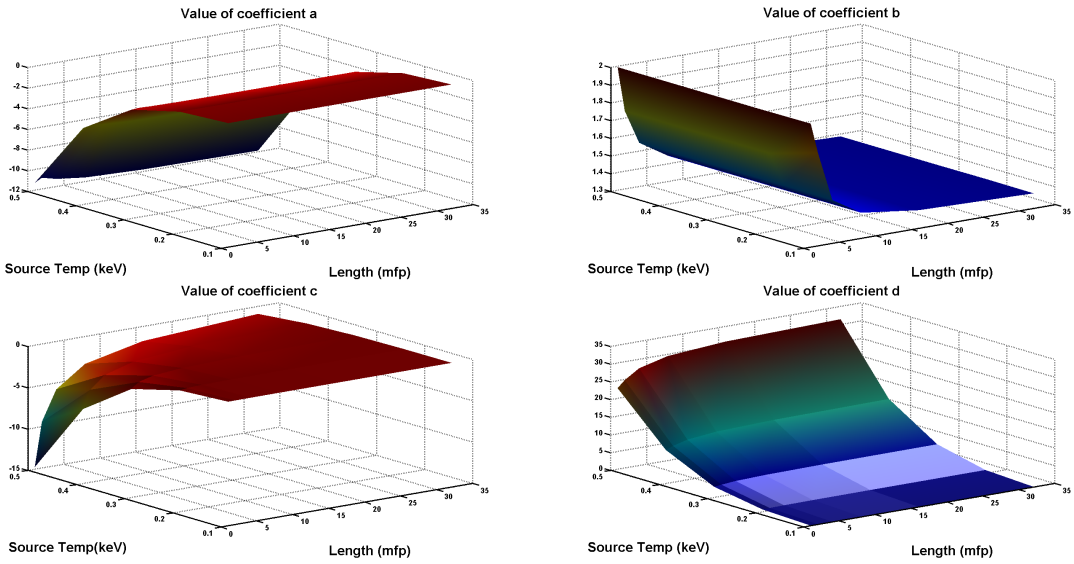


Figure 5: The coefficient surfaces for the photon scalar flux fit function. It can be seen that the coefficients are both a function of the source temperature and the material length.

The form of the scalar photon flux comes naturally from the form of the radiative transfer equations. It can be seen that in the limit of isotropic scattering (several mean free paths into the material), the familiar asymptotic behavior of the diffusion approximation is recovered. It is of the following form:

$$\phi_{fit}(\overline{x}) = ae^{-b\overline{x}} + c\overline{x} + d \tag{40}$$

If the data are plotted in terms of the mean free path ($\overline{x}$), the dependence on the opacity effectively disappears. The functional dependence of the profiles on the source temperature, which we will refer to as T_s, and the material thickness, which we will refer to as ℓ, is not so straightforward. As described above, a parameter scan in which the source temperature and material thickness were simultaneously changed was performed. The resulting fit coefficients, a, b, c and d, for each simulation were plotted and interpolated into a smooth surface. Each surface map is essentially a lookup table.

Therefore, the surrogate functions can be used to generate the scalar flux using the following expression:

$$\phi_{fit}(\overline{x}, T_s, \ell) = a(T_s, \ell)e^{-b(T_s,\ell)\overline{x}} + c(T_s, \ell)\overline{x} + d(T_s, \ell) \tag{41}$$

The user can now specify the parameters which will be used to generate the scalar photon flux. An example surrogate function for a case with a 0.1 keV source temperature and a length of 8 mfps is shown in Figure 6.

Radiation Diffusion Equation in Steady-State:

$$-\frac{d}{dx}\frac{1}{3\Sigma_a(x,T)}\frac{d}{dx}\phi_s + \Sigma_a(x,T)\phi_s = S(\psi_{l_s}) + \Sigma_a acT_s{}^4 \tag{42}$$

Material Internal Energy Equation:

$$\Sigma_a(x,T)\phi_s = \Sigma_a(x,T)acT_s{}^4 \tag{43}$$

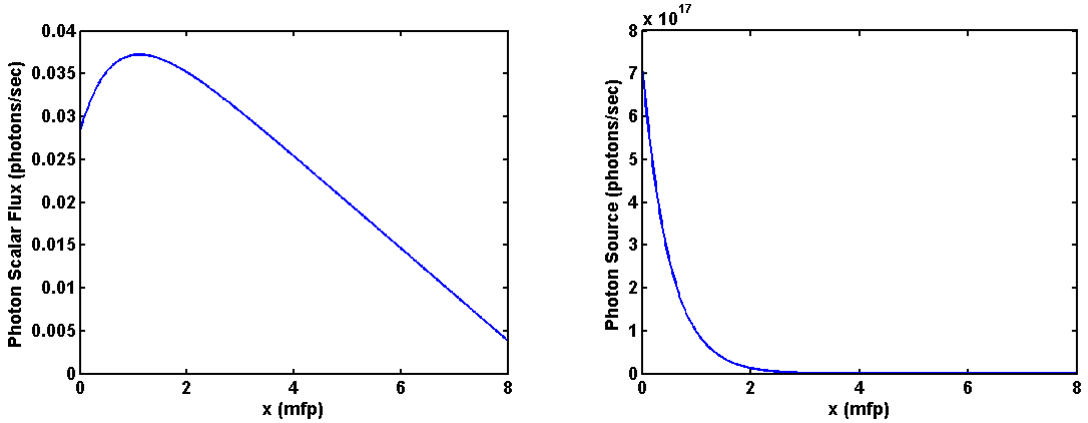


Figure 6: (left) An example profile of scalar photon flux generated by the surrogate function. (right) An example source function determined using the coefficients of the surrogate function.

where the subscript s indicates a stationary quantity. Combining these equations leads to:

$$-\frac{d}{dx}\frac{1}{3\Sigma_a(x,T)}\frac{d}{dx}\phi_s = S(\psi_l) \tag{44}$$

The source term is therefore proportional to the second derivative of the photon scalar flux. Since we have determined an analytic form that described the scalar flux, we can substitute it into this equation to find the form of the new source term. An example source term determined in this manner is shown in Figure 6.

$$S(\overline{x},T_s,\ell) = -\frac{1}{3}\Sigma_a a(T_s,\ell)b(T_s,\ell)^2 e^{-b(T_s,\ell)\overline{x}} \tag{45}$$

Now that the form of the surrogate source function has been determined, it can be inserted directly into a radiation diffusion code. However, with the addition of this new source term, care must be taken to ensure that the boundary conditions are still correct. A form similar to the Marshak boundary condition was used.

The LHS boundary condition (at x=0):

$$J_{in} = \phi_{fit} - \frac{A}{3}\frac{d\phi_{fit}}{dx} \tag{46}$$

The RHS boundary condition (at x=L):

$$0 = \phi_{fit} + \frac{B}{3}\frac{d\phi_{fit}}{dx} \tag{47}$$

Since the analytic form of ϕ_{fit} has been determined, it can be substituted into the boundary conditions to obtain the values of A and B in terms of the fit coefficients a, b, c and d necessary to ensure that the proper conditions are satisfied.

It should be emphasized that the source term is a steady-state function, even though it is used in a time-dependent diffusion calculation. Furthermore, the finite length of the material slab allows the system to settle into an equilibrium that would not be possible in the case of an infinite medium.

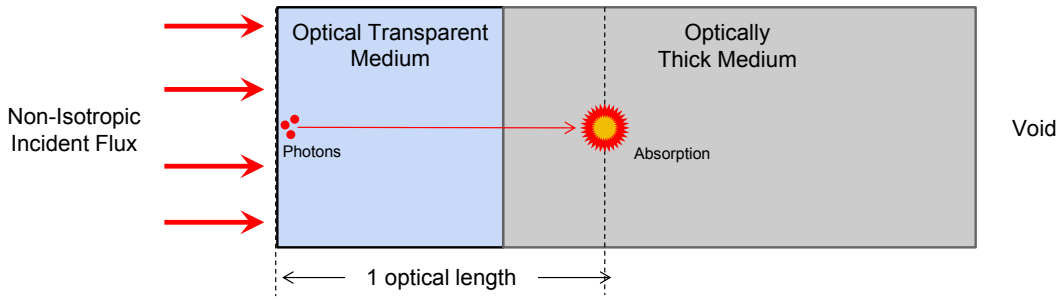


Figure 7: The incoming photons coming from the boundary are replaced by a single isotropic point-source.

8.2 Point-Source Term

It can be shown mathematically that the steady-state solution of the pure diffusion formulation (35-36) will always be monotonically decreasing. Hence, the maximal in scalar flux and temperature shown by the full transport solution could never be obtained by a simple modification of the boundary conditions. Thus, the diffusion formulation must be modified.

One possible solution is to add a photon source term at the place where the maximal in scalar flux is occurring. This approach was introduced earlier by Haskell. [9] Haskell argued that in a semi-infinite geometry with a photon source, which is typically a laser beam normally incident to the surface, most photon absorption events occur at one optical depth inside the material. Hence, he makes the approximation that the incoming photons from the boundary are replaced by a single isotropic point-source located a transport mean free path into the medium. A schematic of the approximation is presented in Figure 7. The formulation of the method is as follows:

Radiation Diffusion Equation:

$$\frac{1}{c}\frac{\partial}{\partial t}\phi(x,t) - \frac{\partial}{\partial x}\frac{1}{3\Sigma_a(x,T)}\frac{\partial}{\partial x}\phi + \Sigma_a(x,T)\phi = \underbrace{S(\psi_l)\delta(x-x_s(t))}_{source\ term} + \Sigma_a acT^4 \tag{48}$$

Material Internal Energy Equation:

$$C_v(x,T)\frac{\partial}{\partial t}T(x,t) = \Sigma_a(x,T)\phi - \Sigma_a(x,T)acT^4 \tag{49}$$

As seen above, the radiation diffusion equation is modified in order to include an isotropic source term of magnitude $S(\psi_l)$ located at the position $x_s(t)$. The magnitude of the source term is a function only of the original incoming photon flux boundary condition. It is equal to the total number of incoming photons in the original situation:

$$S(\psi_l) = \int_0^1 \psi_l(\mu')\,d\mu' \tag{50}$$

The isotropic source term is located at one mean free path into the medium:

$$\int_0^{x_s(t)} \Sigma_a(x', T(x',t))\,dx' \ = 1 \tag{51}$$

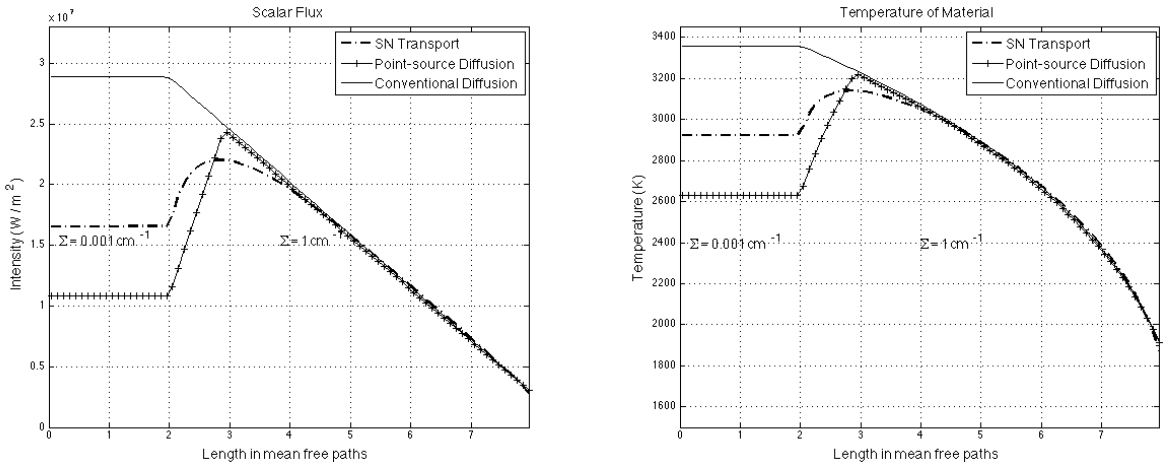


Figure 8: Stationary Scalar Flux and Temperature Fields.

It is worth noting that if the absorption coefficients are temperature dependent, the location of the point-source is time-dependent.

One possible complication that could arise from this method would be to treat the case where there is a non mono-directional incoming flux. One possible solution to this problem would be to decompose the incoming flux in array bins and for each bin, a separate source term could be placed inside the material. Hence, instead of a single source term, a collection of source terms would be placed. The magnitude of each source term would depend on the portion of the incoming photon flux represented by each bin.

Equations (48-51) together with the initial conditions and void boundary conditions define the problem completely. Results for the stationary distributions of the scalar-flux and temperature are shown in Figure 8. The results of the transport method, conventional diffusion and diffusion with point-source are compared for a situation of normally incident photon flux. As seen in Figure 8, the point-source diffusion shows a maximum in scalar flux and temperature at approximately the same location as the transport solution. Also, the point-source diffusion describes the fields correctly at the asymptotic limit. However, the point-source solution shows a sharper profile and it predicts lower-temperatures and scalar fluxes inside the optically thin material.

In order to correct the deficiencies of the point-source method inside the optically thin material, the last described method could be modified in a way that only a certain fraction of the incoming photons from the boundary are replaced by the point-source. We now introduce the parameter α which denotes the fraction of incoming photons replaced by the source term. Evidently, if $\alpha = 1$ we recover the point-source method by Haskell while for $\alpha = 0$, we have the conventional diffusion method. Hence, the magnitude of the source term and the left boundary condition are re-defined as:

$$S(\psi_l) = \alpha \int_0^1 \psi_l(\mu')\, d\mu' \tag{52}$$

$$\phi(0,t) - \frac{\lambda}{\Sigma_a(0,T)} \frac{\partial}{\partial x}\phi(0,t) = 2(1-\alpha) \int_0^1 W(\mu')\psi_l(\mu')\, d\mu' \quad , \forall t > 0 \tag{53}$$

The results are shown in Figures 9 and 10. The result in Figure 9 differs from that of Figure 10 in the thickness of the slab. The parameter α is varied from zero to one and the resulting stationary

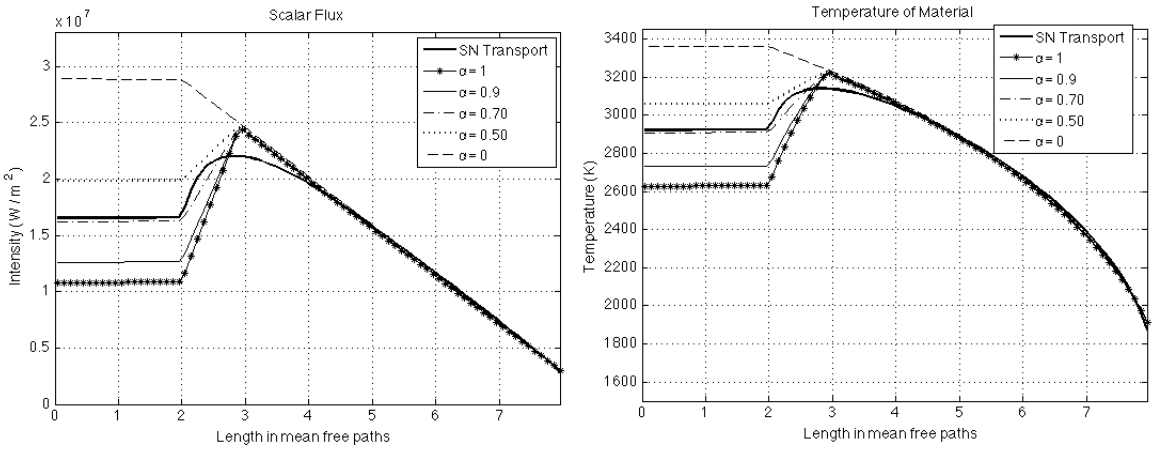


Figure 9: Stationnary Scalar Flux and Temperature Fields for varying α.

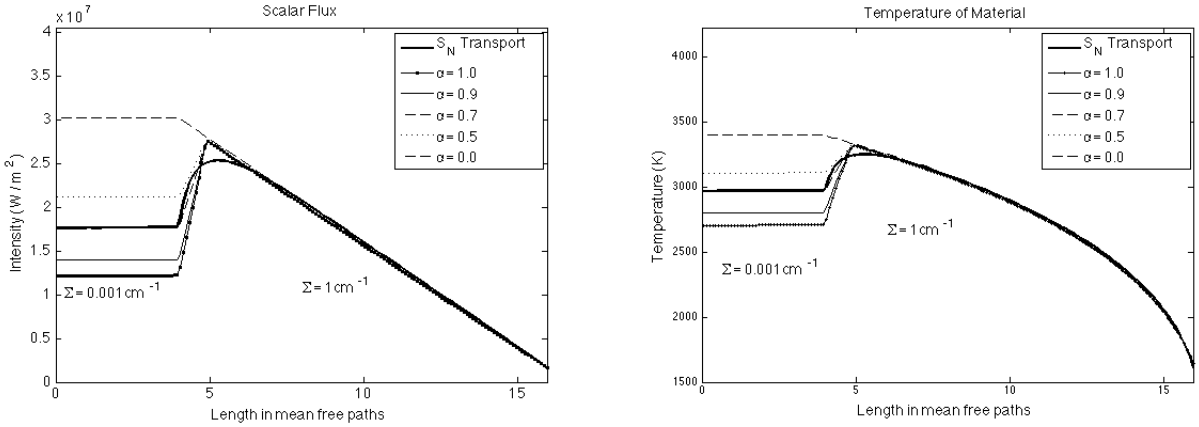


Figure 10: Stationnary Scalar Flux and Temperature Fields for varying α.

scalar flux and temperature fields are plotted. For $\alpha = 1$, the result corresponds to the point-source diffusion method while $\alpha = 0$ recovers the conventional diffusion result. For both geometries, when $\alpha \simeq 0.70$ the point-source method predicts the scalar flux and temperature fields more accurately for the optically thin material. It is speculated that this value for α might be related to the diffusion-limit boundary conditions' constant introduced by Habetler. [12] However, this is still speculation and has not been proven yet. It can also be shown mathematically that the variation of the α does not change the location and magnitude of the maximal values of the fields.

In conclusion, improved results were obtained when 70% of the incoming photon flux was replaced by a point-source located one mean free path inside the medium. It is still unknown if this partitioning is indeed universal or if it is dependent of the system geometry. Future work will include more studies on this method.

8.3 Hybrid Transport-Diffusion Method

In this section we present a hybrid method to model equations (4-9). The method is based on a first-collision approximation. The motivation for this method comes from the following: among the many ways to derive the diffusion approximation from the transport model, one is obtained by assuming that the specific intensity $\psi(x,\mu,t)$ can be represented via a Taylor series to the first order on the angular dependence:

$$\psi(x,\mu,t) = \frac{1}{2}\phi(x,t) + \frac{3}{2}\mu J(x,t) + \mathcal{O}(\mu^2)$$

Hence, it is assumed that the specific intensity has a small angular dependency: $|\frac{J}{\phi}| \ll 1$. However, when dealing with normally incident radiation, in which *"transport"* effects are most noticeable, the ratio between the scalar flux and current flux is $|\frac{\phi}{J}| \simeq 1$. Hence, the angular expansion shown above is invalid. The diffusion approximation cannot model the streaming photon radiation correctly.

In order to solve this problem, the radiation field can be divided into two parts: the first part represents the uncollided photon flux while the latter represents the photons emitted by the material. This decomposition has been studied before in neutron transport theory. [13,14,19,20] The decomposition of the radiation flux would be written as follows:

$$\psi(x,\mu,t) = \psi_{uncollided}(x,\mu,t) + \psi_{emitted}(x,\mu,t) \tag{54}$$

where the uncollided flux and the emitted flux satisfy:

$$\frac{1}{c}\frac{\partial}{\partial t}\psi_{uncollided} + \mu\frac{\partial}{\partial x}\psi_{uncollided} + \Sigma_a\psi_{uncollided} = 0 \tag{55}$$

$$\frac{1}{c}\frac{\partial}{\partial t}\psi_{emitted} + \mu\frac{\partial}{\partial x}\psi_{emitted} + \Sigma_a\psi_{emitted} = \frac{1}{2}\Sigma_a acT^4 \tag{56}$$

At the continuum level, this splitting is exact, i.e., if $\psi_{uncollided}$ solves (55) and $\phi_{emitted}$ satisfies (56), then ψ will satisfy (1) (with appropriate boundary conditions). Note that the equation for $\psi_{uncollided}$ is an equation for a pure absorber: there is no coupling in angle. Moreover, it is independent of $\psi_{emitted}$. The angular integral of $\psi_{uncollided}$ will act as a (first-collision) source for the internal energy equation.

The original idea of this type of splitting is to evaluate $\psi_{uncollided}$ using an S_N transport method. Since the source term of equation (56) is isotropic, the emitted radiation can be modeled by using a simple diffusion approximation. To the best of our knowledge, such a decomposition of the transport-diffusion for thermal radiative transfer has never yet been explored. The formulation of the method is as follows:

Uncollided Radiation Transport

$$\frac{1}{c}\frac{\partial}{\partial t}\psi_{uncollided} + \mu\frac{\partial}{\partial x}\psi_{uncollided} + \Sigma_a\psi_{uncollided} = 0 \tag{57}$$

Emitted Radiation Diffusion

$$\frac{1}{c}\frac{\partial}{\partial t}\phi_{emitted} - \frac{\partial}{\partial x}\frac{1}{3\Sigma}\frac{\partial}{\partial x}\phi_{emitted} + \Sigma_a\phi_{emitted} = \Sigma_a acT^4 \tag{58}$$

Material Internal Energy

$$C_v(x,T)\frac{\partial}{\partial t}T = \Sigma_a\left\{\phi_{diff} + \int_{-1}^{1}\psi_{uncollided}(x,\mu',t)\,d\mu'\right\} - \Sigma_a acT^4 \tag{59}$$

While the initial conditions are trivially obtained from the original statement of the problem, the boundary conditions are modified and adapted as follows:

$$\psi_{uncollided}(x=0,\mu,t) = \psi_l(\mu), \forall\mu \in [0,1], \forall t > 0 \tag{60}$$

$$\psi_{uncollided}(x=L,\mu,t) = 0, \forall\mu \in [-1,0], \forall t > 0 \tag{61}$$

$$\phi_{emitted}(0,t) - \frac{\lambda}{\Sigma_a(0,T)}\frac{\partial}{\partial x}\phi_{emitted}(0,t) = 0, \forall t > 0 \tag{62}$$

$$\phi_{emitted}(L,t) + \frac{\lambda}{\Sigma_a(L,T)}\frac{\partial}{\partial x}\phi_{emitted}(L,t) = 0, \forall t > 0 \tag{63}$$

where the conventional transport boundary conditions are adopted for the uncollided photons and the asymptotic diffusion limit boundary conditions are used for the emitted photons. [12]

8.4 Hybrid Semi-Stationary Method

The Hybrid model, described above, requires a transport equation solver for the uncollided flux. Although such a solver is quite simple, some large radiation hydrocodes do not include a transport module and only have diffusion solvers. Since one of the main objectives of this work is to develop a new diffusion model that captures the *"transport effects"* and that can be implemented into the rad-hydro code FLAG, it is advantageous to eliminate the need of using a transport solver in the hybrid method.

It is possible to simply equations (57-59) by assuming the non-retardation approximation on the uncollided radiation field. This approximation suggests that $c \to +\infty$ for the uncollided photons in equation (57). Physically, the flight time of the uncollided photons is ignored. The modified equation will be hence written as:

$$\cancelto{0}{\frac{1}{c}\frac{\partial}{\partial t}\psi_{uncollided}} + \mu\frac{\partial}{\partial x}\psi_{uncollided} + \Sigma_a\psi_{uncollided} = 0 \tag{64}$$

In this model, the transport equation for the uncollided photon flux is solved without the temporal derivative; hence, the name of the model. It is important to notice that although the temporal derivative is ignored, the uncollided photon flux can still be time dependent. This occurs in the case in which the photon absorption coefficients Σ_a are temperature dependent.

Although the above approximation might be considered too aggressive and its validity is somewhat questionable, it is believed that the errors introduced by the non-retardation approximation are small. Firstly, the location where the biggest error might be introduced is away from the incoming flux boundary. The non-retardation approximation predicts that at a time $t = 0^+$, the uncollided scalar radiation flux is present everywhere in the system. Although the error is largest at locations farthest from the boundary, the uncollided scalar flux also decreases exponentially with the optical mean free path. Hence, there might be a compensation effect. A second reason why this approximation would not introduce large errors is that the resulting scalar and temperature wave (Marshak wave) that results from the configuration in this work originates from a feedback between the emitted radiation and the material temperature. Since in steady-state the portion of uncollided radiation is negligible compared to the emitted radiation, this leads to the belief that the non-retardation approximation would not change the results abruptly.

The complete formulation for the Hybrid Semi-Stationary model is as follows: *Uncollided Radiation Field*

$$\psi_{uncollided}(x,\mu,t) = \psi_l(\mu) exp\left\{-\int_0^x \Sigma_a(x', T(x',t))\frac{1}{\mu}\,dx'\right\} \tag{65}$$

Emitted Radiation Diffusion

$$\frac{1}{c}\frac{\partial}{\partial t}\phi_{emitted} - \frac{\partial}{\partial x}\frac{1}{3\Sigma}\frac{\partial}{\partial x}\phi_{emitted} + \Sigma_a\phi_{emitted} = \Sigma_a acT^4 \tag{66}$$

Material Internal Energy

$$C_v(x,T)\frac{\partial}{\partial t}T = \Sigma_a\left\{\phi_{diff} + \int_{-1}^{1}\psi_{uncollided}(x,\mu',t)\,d\mu'\right\} - \Sigma_a acT^4 \tag{67}$$

in which the uncollided radiation field $\psi_{uncollided}(x,\mu,t)$ is solved from equation (64). The initial and boundary conditions for the Hybrid Semi-Stationary model remain the same as those presented for the Hybrid model (60-63).

9 Results

The results of the methods described in Section 8 will be discussed here. The surrogate source model will be discussed first and compared to the S_N method, the method we assume provides the correct answer. The remaining three methods will be discussed afterwards with an emphasis on the difference in performance in cases with constant and non-constant heat capacity.

9.1 Results of the Surrogate Source Term Method

As described above, the surrogate source term is inherently an empirical method. Large-scale parameter scans were performed with the goal of distilling down the large amounts of data to a concise physical model. A fit function, which we call a surrogate function, informed by the structure of the radiative transfer equations, was used to fit the photon scalar flux. The behavior of the four fit coefficients was investigated as a function of opacity, source temperature, and material length. The results were used to generate a surrogate source term which was implemented in a diffusion code to evaluate any possible improvements to the conventional diffusion model.

Using the appropriate boundary conditions and the steady-state surrogate source term, the radiation source diffusion calculation was run and compared to the S_N solution, the steady-state solution (calculated using the surrogate function), and the standard diffusion model. These results are shown in Figure 11 at three different time-steps. It can be seen that the surrogate source model remains similar to the S_N solution throughout the simulation. In steady-state, the difference between the two models for the peak material temperature is approximately half a percent. This agreement is promising and suggests that this method is ready for more rigorous testing and implementation in a larger code such as FLAG.

It should be noted that the results presented are for only a single case. Future work will include investigating additional problems that include: dissimilar opacities, non-constant heat capacity, real materials, and time-dependent behavior. If the method continues to provide good results, a new module will be written and incorporated into the FLAG code to try to improve the accuracy of rad-hydro calculations at a material interface.

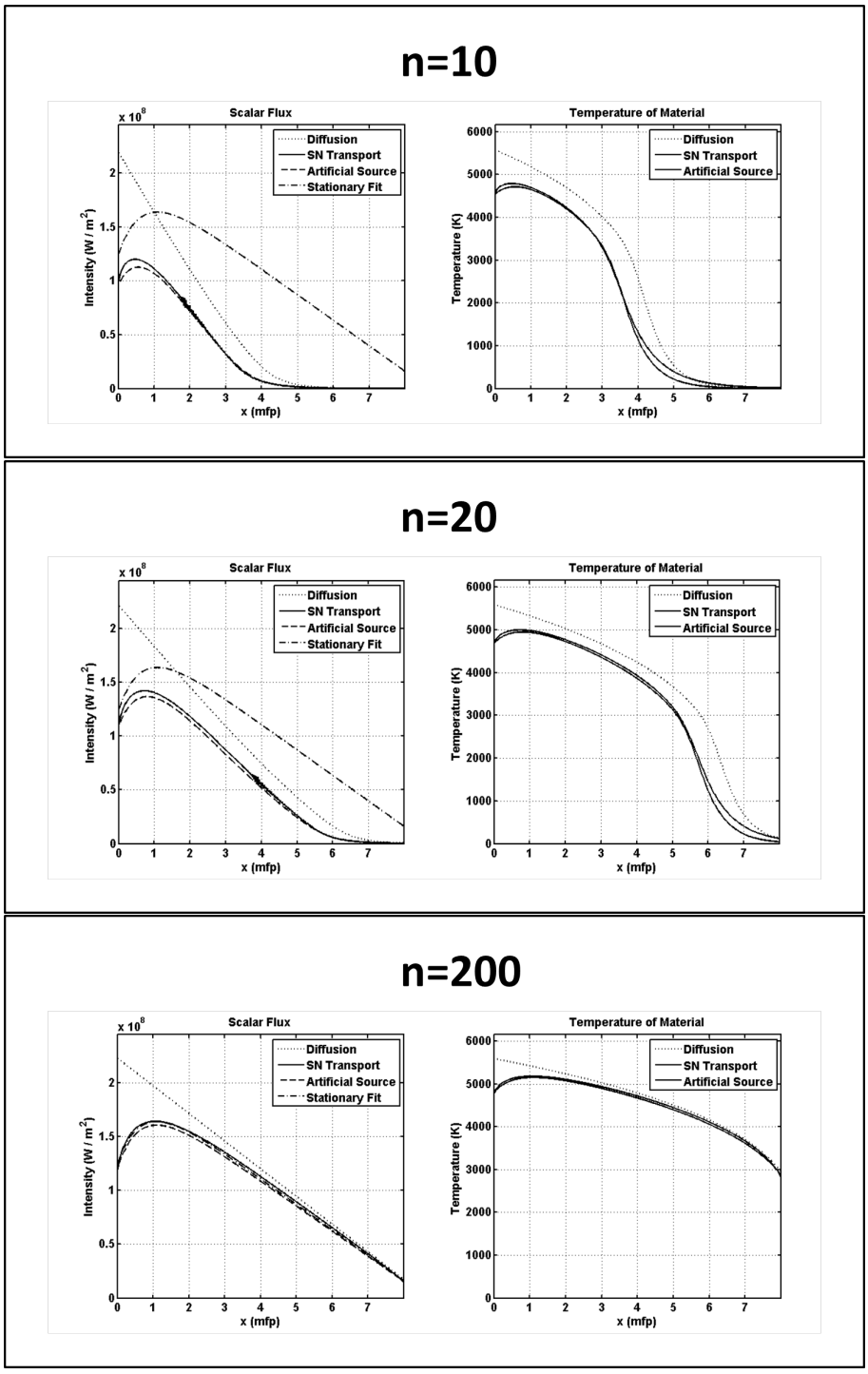


Figure 11: Comparison of the intensities from several methods: diffusion, diffusion with the surrogate source term, S_N, and the steady-state solution. Each plot at three different time-steps (n=10,n=20, and n=200) shows the evolution of scalar photon flux (left) and the material temperature (right). It can be seen that the surrogate source term method agrees very well with the S_N results.

9.2 Results of the Point-Source, Hybrid-Diffusion, and Hybrid Semi-Stationary Methods

In this section, a comparison is made between the methods described in Section 8. The model to which the other methods will be compared is the $S_{N=20}$ transport model. The objective will be to see how the methods previously described methods improve the conventional diffusion model.

Two cases will be studied in this section. These are:

- In the first case, a normally incident beam of photons incident from the left boundary travels through an optically thin and thick media. The media have the same microscopic characteristics (same σ_a and c_v). Only their densities vary; the density of the optically thin material is 1000 times smaller. The boundary between the two materials is located at 2mfp. The initial temperature and initial scalar flux are zero.
- In the second case, the only change is that the absorption cross-sections are temperature-dependent. They follow a law of the following form:

$$\Sigma_a(x,T) = \Sigma_{a,0}(x)(\frac{T_{ref}}{T})^3$$

  In order to avoid any singularities associated with zero temperature, the initial temperature is 1000K. The reference temperature T_{ref} is 1000K.

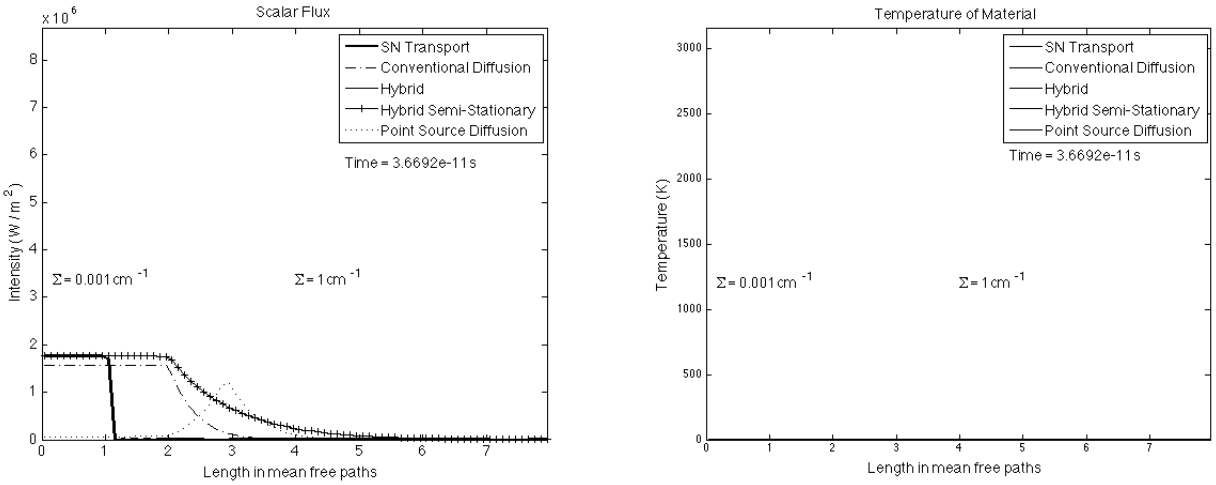
9.2.1 Comparison with constant Σ_a

The results for the first case are shown in Figure 12. The plots in Figure 12 show the temporal evolution of the scalar flux and the temperature fields. The scalar flux is plotted on the left and the material temperature is plotted on the right. The temporal sequence increases from top to bottom.

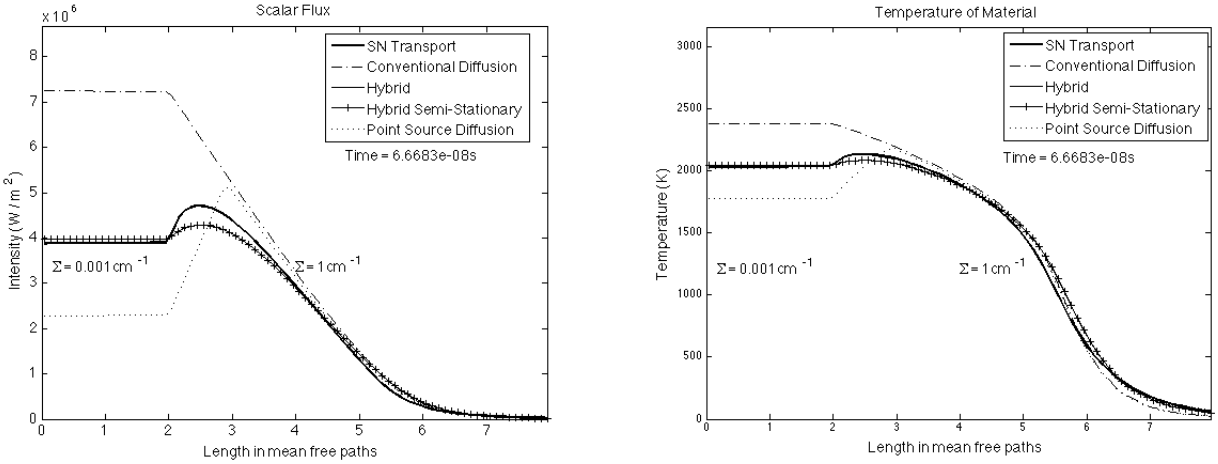
In the first time-frame, some of the main differences between the methods are noticeable. As it can be seen, the photon wave front is shown on the left side. It is accurately described by the S_N method and the Hybrid method. The diffusion solution has already propagated through the thin medium because a constant Eddington factor was used in this study. If a flux-limited diffusion had been used, the diffusion model would describe the wavefront more accurately. [6] The Hybrid Semi-Stationary model predicts that the scalar flux has already propagated through all the medium. This is expected since the non-retardation approximation was adopted for the uncollided radiation field. Finally, the Point-Source diffusion predicts a strong radiation field at around 3 mfp due to the presence of the photon source.

In the second time-frame, the Marshak wave can be observed. The Marshak wave is a well-known physical phenomenon that has been studied both numerically and analytically. [2, 17, 22] The diffusion model over-predicts higher scalar flux and temperatures inside the optically thin material. The Point-Source diffusion underpredicts the same quantities for the same material. Interestingly, both methods follow the same asymptotic limit inside the thick material. For the Hybrid methods, both predict similar solutions for the fields. Most importantly, both methods capture the *"transport effects"* at the material interfaces. However, their maximum scalar fluxes and temperatures are smaller than those predicted by the S_N method. It is believed that the underestimated scalar flux and temperature inside the thick material is compensated for by the slightly higher scalar flux and temperature inside the thin material.

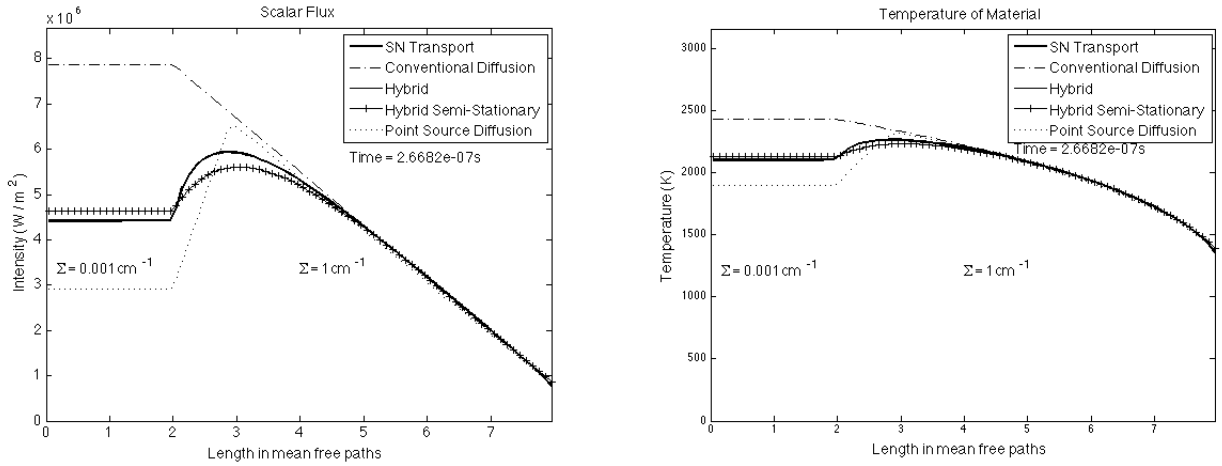
The final time-frame shows the stationary scalar flux and temperature fields. As shown, the diffusion model overestimates the scalar flux and temperature inside the optically thin material. However, it accurately describes the aforementioned quantities at the asymptotic limit. The Point-Source diffusion



(a) Scalar Flux and Temperature Fields at initial time-step.



(b) Scalar Flux and Temperature Fields at time $t = 1.66\ (10)^{-8}$s.



(c) Scalar Flux and Temperature Fields at steady state.

Figure 12: Temporal evolution of Scalar Flux and Temperature Field with constant photon absorption cross-sections.

overestimates the maximal scalar flux and temperature. The methods that best approximate the S_N transport solution are the Hybrid methods. The Hybrid and Hybrid Semi-Stationary predict the same stationary fields. This provides some evidence about the validity of the non-retardation approximation.

As a final note, in this case the relative difference between the predicted maximal temperatures of the S_N and Hybrid solutions was only 3%. Although this difference might not be small enough from a radiation transport perspective, it is still quite promising. If such Hybrid radiation models were coupled with a material hydrodynamics code, the error of the Hybrid methods would not be very significant from the hydrodynamic perspective. Hence, it is very probable that the Hybrid methods would lead to better rad-hydro predictions.

9.2.2 Comparison with Temperature Dependent Σ_a

The results for the second case with temperature dependent Σ_a are shown in Figure 13. The graphs in Figure 13 show the temporal evolution of the scalar flux and the temperature fields. On the left column, the scalar flux is plotted while on the the right column the temperature field is shown. Time increases from top to bottom.

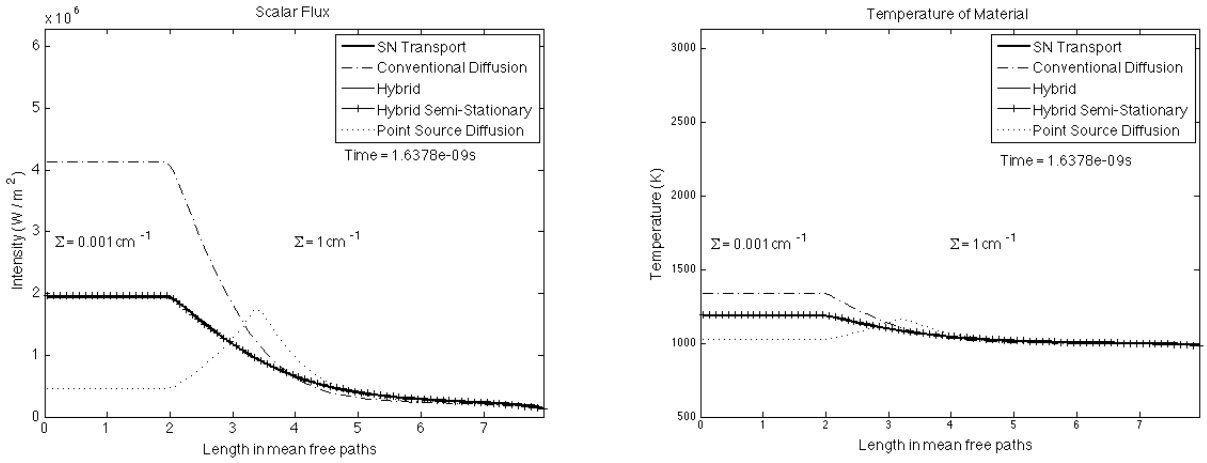
In the first time-frame, the diffusion solution already predicts higher scalar flux and temperatures in the optical thin material. The Point-Source diffusion method shows a maximum scalar flux and temperature at the location of the photon isotropic source. As it can be seen, the Hybrid methods both match the solution given by the S_N method.

In the second time-frame, the photon source has moved to the right at approximately 3 mfp inside the thick material. As shown in equation (51), the source location is defined at one optical depth inside the system. Since the temperature field has increased considerably, the absorption coefficient $\Sigma_a(x,T)$ has decreased. Hence, the photon source needs to move to the right. For the Hybrid methods, the methods underestimate the scalar flux and temperature field inside the system.

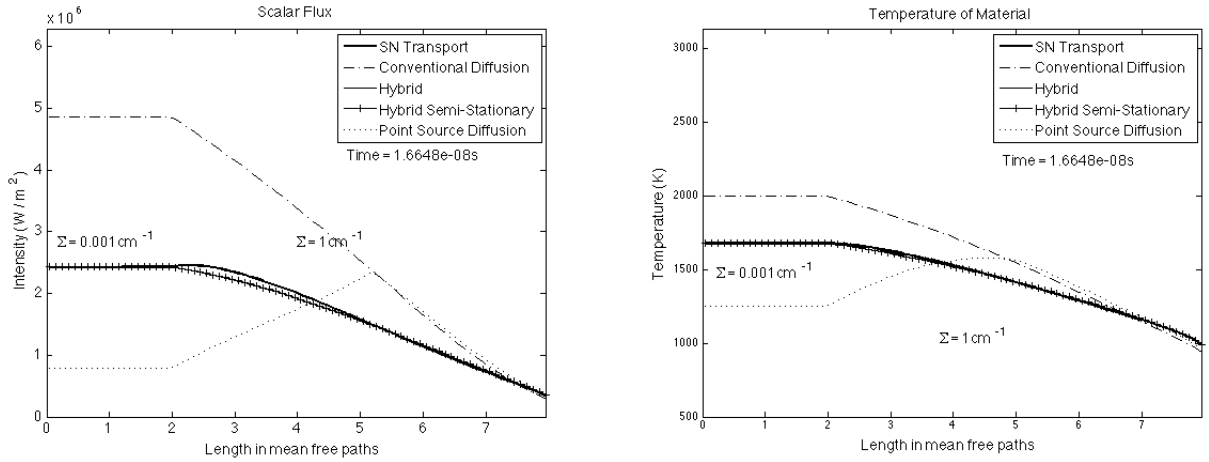
The final time-frame shows the stationary scalar flux and temperature fields. As shown, the diffusion model clearly overestimates the scalar flux and temperature at all points. Surprisingly, the diffusion model no longer predicts correctly the asymptotic limit inside the thick material as in the first case. There are several possible explanations for this effect. The first is that these diverging results are due to the high non-linearity introduced to the problem when the absorption cross-sections were allowed to vary with temperature. It is speculated that some non-linear bifurcation could have produced this error. Another possible explanation might be that there is an insufficient time-implicitness. Hence, not all effects are being captured. More research will have to be done on the subject.

The Point-Source diffusion method clearly gives the wrong answer. Due to the rise in temperature inside the system, the photon source was shifted to the right. It is worth noting that if a simulation with higher incoming photon flux were to be performed, the temperatures inside the material would rise more. Hence, the photon source could ultimately reach the right boundary of the system. In this situation, the Point-Source diffusion model would eventually collapse. However, if the Point-Source reaches the right boundary in the simulations, the material has become too thin and the diffusion approximation would become invalid anyways.

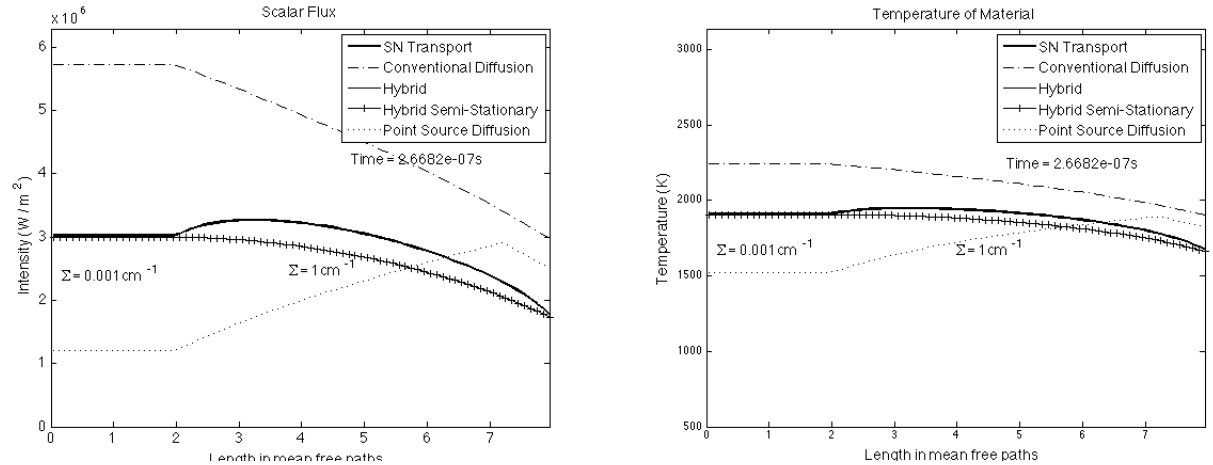
As shown, the Hybrid methods evidently improve the solution using the conventional diffusion approach. However, they do not accurately reproduce the solution given by the transport S_N method as in the first case. Although the results might be discouraging, it is important to note that in this numerical simulation, the temperature rose by almost a factor of two in the entire system. Hence, the photon mean free path increased almost 8 times in the thick material. Since the photon mean free path increased to a value comparable to the length of the system, it is expected that the diffusion approximation used



(a) Scalar Flux and Temperature Fields at time $t = 1.64(10)^{-9}$s.



(b) Scalar Flux and Temperature Fields at time $t = 1.66(10)^{-8}$s.



(c) Scalar Flux and Temperature Fields at steady state.

Figure 13: Temporal evolution of Scalar Flux and Temperature Field with temperature dependent absorption cross-sections.

in the Hybrid methods would eventually collapse because there are no longer sufficient collisions. It is expected that for situations in which the final photon mean free path is smaller than the system's length, the Hybrid methods would give correct results.

10 Implementation in FLAG

In this section, preliminary investigation into the mechanics of implementing the previous methods in FLAG will be discussed. The equations of grey radiative hydrodynamics under the assumption of local thermodynamic equilibrium can be found in Mihalas and Mihalas. [16] We have chosen to use the fluid (or co-moving) frame formulation:

$$\frac{\partial}{\partial t}\rho + \underline{\nabla} \cdot (\rho \underline{v}) = 0 \tag{68}$$

$$\frac{\partial}{\partial t}(\rho \underline{v}) + \underline{\nabla} \cdot (\rho \underline{v} \otimes \underline{v}) = -\underline{\nabla P} + \frac{1}{c}\Sigma_a \underline{F_r} \tag{69}$$

$$\frac{\partial}{\partial t}(\rho e) + \underline{\nabla} \cdot (\rho e \underline{v}) = -\Sigma_a acT^4 + \Sigma_a cE_r \tag{70}$$

$$\frac{\partial}{\partial t}(E_r) + \underline{\nabla} \cdot (E_r \underline{v}) = -\underline{\nabla} \cdot \underline{F_r} - \underline{\underline{P_r}} : \underline{\underline{\nabla v}} + \Sigma_a acT^4 - \Sigma_a cE_r \tag{71}$$

$$\frac{\partial}{\partial t}(\underline{F_r}) + \underline{\nabla} \cdot (\underline{F_r} \otimes \underline{v}) = -c^2 \underline{\nabla} \cdot \underline{\underline{P_r}} - c\Sigma_a \underline{F_r} \tag{72}$$

where we have used the conventional notation used in the radiation hydrodynamics literature. The first three equations describe the conservation of mass, momentum and internal energy of the material. The last two equations describe the evolution of the radiation field.

The variables for the material properties are: ρ the matter density, v the velocity, e the matter internal energy density, P and T the pressure and the temperature of the material, and Σ_a the absorption opacity. The radiation field variables are: E_r is the radiative energy density, F_r is the radiative flux, and P_r is the radiative pressure tensor . The radiation field variables are defined as:

$$E_r = \frac{1}{c}\int_0^{+\infty}\int_{4\pi} \psi d\Omega d\nu$$

$$\underline{F_r} = \int_0^{+\infty}\int_{4\pi} \underline{\Omega}\ \psi\ d\Omega\ d\nu$$

$$\underline{\underline{P_r}} = \frac{1}{c}\int_0^{+\infty}\int_{4\pi} \underline{\Omega} \otimes \underline{\Omega}\ \psi\ d\Omega\ d\nu$$

In the previous section, several methods were explored in order to improve the radiation diffusion calculations. Here, only the Hybrid Semi-Stationary Method will be implemented into the radiation hydrodynamic equations. However, the team hopes to explore the implementation of the other described methods.

In the Hybrid Semi-Stationary Model, the radiation field is decomposed into two parts:

$$\psi = \psi_u + \psi_e$$

where ψ_u describes the radiation field composed by the uncollided photons while ψ_e represents the emitted photons by the material. The validity of this decomposition was discussed in the previous section.

The equations that are modified are the following:

$$\frac{\partial}{\partial t}(\rho \underline{v}) + \underline{\nabla} \cdot (\rho \underline{v} \otimes \underline{v}) = -\underline{\nabla P} + \frac{1}{c}\Sigma_a \left\{ \underline{F_c} + \underline{F_e} \right\} \tag{73}$$

$$\frac{\partial}{\partial t}(\rho e) + \underline{\nabla} \cdot (\rho e \underline{v}) = -\Sigma_a acT^4 + \Sigma_a c \left\{ E_u + E_e \right\} \tag{74}$$

$$\frac{\partial}{\partial t}(E_e) + \underline{\nabla} \cdot (E_e \underline{v}) = -\underline{\nabla} \cdot \underline{F_e} - \underline{\underline{P_e}} : \underline{\underline{\nabla v}} + \Sigma_a acT^4 - \Sigma_a cE_e \tag{75}$$

$$\frac{\partial}{\partial t}(\underline{F_e}) + \underline{\nabla} \cdot (\underline{F_e} \otimes \underline{v}) = -c^2 \underline{\nabla} \cdot \underline{\underline{P_e}} - c\Sigma_a \underline{F_e} \tag{76}$$

The previous equations describe the the dynamics of the material properties and of the emitted radiation field. In order to describe the uncollided radiation field, ψ_u, we will use the frequency-dependent co-moving frame transport equation which can be found in Castor: [5]

$$\frac{1}{c}\left\{ \frac{\partial}{\partial t}\psi_{u,\nu} + \underline{v} \cdot \underline{\nabla \psi_{u,\nu}} \right\} + \underline{\Omega} \cdot \underline{\nabla \psi_{u,\nu}} = \frac{\nu}{c}\underline{\Omega} \cdot \underline{\underline{\nabla v}} \cdot \underline{\nabla_{\nu \underline{\Omega}} \psi_{u,\nu}} - \frac{3}{c}\underline{\Omega} \cdot \underline{\underline{\nabla v}} \cdot \underline{\Omega} \psi_{u,\nu} - \Sigma_a \psi_{u,\nu} \tag{77}$$

As it can be seen, the transport equation in the co-moving frame is more complicated than that in the fixed-laboratory frame. From a mathematical point of view, the co-moving transport equation is a PDE for one scalar dependent variable in seven independent variables: x, y, z, two angles for Ω and t. By contrast, the fixed-frame equation has only four independent variables: x, y, z and t. The three photon momentum coordinates are only parameters. [5] In the following, the co-moving transport equation will be solved in a simplified case.

The 1-D radiation problem was the focus of this work. Hence, we will define the spatial coordinate x and use azimuthal symmetry. As done previously, we will apply the non-retardation approximation so that the photon flight time can be neglected. The aberration correction in equation (77) will be neglected. The simplified equation to solve is: [5]

$$(\mu + \frac{v}{c})\frac{\partial}{\partial x}\psi_{u,\nu} = -\frac{1}{c}\mu^2 \frac{\partial v}{\partial x}\ \nu \frac{\partial}{\partial \nu}\psi_{u,\nu} - \Sigma_a \psi_{u,\nu} \tag{78}$$

This last equation is still frequency-dependent. We can integrate in frequency in order to consider only the grey transport equation:

$$(\mu + \frac{v}{c})\frac{\partial}{\partial x}\psi_u = -\mu^2 \frac{1}{c}\frac{\partial v}{\partial x}\psi_u - \Sigma_a \psi_u \tag{79}$$

This last equation can be easily solved. We note $\psi_l(\mu, t)$ the value of the incoming flux at the left boundary (x=0) of the physical domain. The solution is:

$$\psi_u(x, \mu, t) = \psi_l(\mu, t)\ exp\left\{ -\int_0^x \frac{1}{v(x', t) + \mu c}\left(\frac{\partial v}{\partial x'}(x', t) + c\Sigma_a(x', T(x', t)) \right) dx' \right\} \tag{80}$$

The last equation is the solution for the uncollided photon flux traveling through a fluid with velocity field $v(x, t)$ in a 1D semi-infinite geometry. An interesting remark to consider is that if $\mu c \gg v$, the

solution above takes the form of the solution for the uncollided flux traveling through a stationary, fixed material:

$$\psi_u(x,\mu,t) = \psi_l(\mu,t)\ exp\left\{-\int_0^x \Sigma_a(x', T(x',t))\frac{1}{\mu}dx'\right\}$$

As an example, let $\psi_l(\mu,t) = \psi_0\delta(\mu-1)$. This represents the case of a normally incident photon beam on a material surface. The uncollided radiative energy density and uncollided radiative flux are:

$$E_u = \frac{1}{c}\psi_0\ exp\left\{-\int_0^x \frac{1}{v(x',t)+c}\left(\frac{\partial v}{\partial x'}(x',t) + c\Sigma_a(x',T(x',t))\right)dx'\right\} \tag{81}$$

$$\underline{F_u} = cE_u\underline{e}_x \tag{82}$$

where $\underline{e}_x$ is the unit vector in the x direction.

Now let's turn our attention to the emitted radiation field. As done previously, we adopt the diffusion approximation for the emitted radiation field. Hence, we provide the closures:

$$\underline{F_e} = -\frac{c}{3\Sigma_a}\underline{\nabla E_e} \quad \underline{\underline{P_e}} = \frac{1}{3}E_e\underline{\underline{1}}$$

The last closures are inserted into equations (73-76). Since only the 1D problem is treated in this work, the resulting equations are the following:

$$\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho v^2) = -\frac{\partial}{\partial x}P - \frac{1}{3}\frac{\partial}{\partial x}E_e + \frac{1}{c}\Sigma_a F_u \tag{83}$$

$$\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial x}(\rho e v) = -P\frac{\partial v}{\partial x} + \Sigma_a c\left\{E_e + E_u - aT^4\right\} \tag{84}$$

$$\frac{\partial}{\partial t}E_e - \frac{\partial}{\partial x}\frac{1}{3\Sigma(x,T)}\frac{\partial}{\partial x}E_e + \frac{4}{3}\frac{\partial}{\partial x}(E_e v) = \Sigma_a c\left\{aT^4 - E_e\right\} + \frac{1}{3}v\frac{\partial E_e}{\partial x} \tag{85}$$

These equations can be then put to Lagrangian form by using the continuity equation (68). Hence, the complete system of equations is:

$$\rho\frac{D}{Dt}v = -\frac{\partial}{\partial x}P - \frac{1}{3}\frac{\partial}{\partial x}E_e + \frac{1}{c}\Sigma_a F_u \tag{86}$$

$$\rho\frac{D}{Dt}e = -P\frac{\partial v}{\partial x} + \Sigma_a c\left\{E_e + E_u - aT^4\right\} \tag{87}$$

$$\rho\frac{D}{Dt}\left(\frac{E_e}{\rho}\right) - \frac{\partial}{\partial x}\frac{1}{3\Sigma(x,T)}\frac{\partial}{\partial x}E_e = +\Sigma_a c\left\{aT^4 - E_e\right\} - \frac{1}{3}E_e\frac{\partial v}{\partial x} \tag{88}$$

together with equations (81-82) for the uncollided radiation field.

The team hopes that these equations could be implemented into the Exa-Flag testbed in order to test the validity and advantages of the "Hybrid Stationary Method" in a radiation-hydrodynamics context. Additional research on the implementation of the other methods described in Section 8 will be performed in the future.

11 Conclusions and Future Work

In this work, boundary physics of thermal radiation transport were investigated using full-transport methods and high-fidelity diffusion methods. In the first part of this work, the physics at a material interface was characterized using a large-scale parameter scan. With the insight gained from these studies, different diffusion-based methods were developed in order to improve conventional diffusion predictions.

These methods were divided into two categories: the first one was based on empirical methods and the second one was based on analytic methods. The first method used the information obtained in the parameter scans to develop a surrogate source term that would improve the diffusion solution to be closer to the transport prediction. The second category of methods were based on analytic, physical models that have not been yet applied to thermal radiation transport. While the results of the surrogate method were closer to the transport results, it is constrained by the parameter scan on which it is based. The analytic models are more general; however their results are not as close to the transport solution but still capture the main *"transport effects"*.

Future work may include the implementation of the above methods into a radiation hydrodynamics calculation. The team hopes that the material hydrodynamic predictions will be improved by the developed diffusion methods, which can capture the *"transport effects"* at the material interfaces.

References

[1] G. Allaire and F. Golse. *Transport et Diffusion.* Ecole Polytechnique, Paris, 2012.

[2] G.C. Pomraning B.D. Ganapol. The non-equilibrium marshak wave problem: A transport theory solution. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 29:311–320, 1983.

[3] K. G. Budge. *A Users' Guide to the Serrano Testbed.* Los Alamos National Lab.

[4] Keneth Case. Recent developments in neutron transport theory. *The University of Michigan*, 1961.

[5] John Castor. *Radiation Hydrodynamics.* Lawrence Livermore National Laboratory, California, 2003.

[6] G.C. Pomraning C.D. Levermore. A flux-limited diffusion theory. *Astrophys. J.*, 248:321, 1981.

[7] G.L. Olson D.A. Knoll, W.J. Rider. An efficient nonlinear solution method for non-equilibrium radiation diffusion. *J. Quant. Spect. Rad. Trans.*, 63:15, 1999.

[8] C. Baldwin et al. Iterative linear solvers in a 2d radiation-hydrodynamics code: methods and performance. *J. Comput. Phys.*, 154:1, 1999.

[9] R.C. Haskell et al. Boundary conditions for the diffusion equation in radiative transfer. *JOSA A*, 16,10:2727–2741, 1994.

[10] J. A. Fleck and J. D. Cummings. An implicit monte carlo scheme for calculating time and frequency dependent nonlinear radiation transport. *J. Comp. Phys.*, 8:313–342, 1971.

[11] S. Glasstone G.I. Bell. *Nuclear Reactor Theory.* Krieger Publishing, Florida, 1985.

[12] G.J. Habetler and B.J. Matkowsky. Uniform asymptotic expansions in transport theory with small mean free paths, and the diffusion approximation. *Journal of Mathematical Physics*, 16:846, 1975.

[13] C. Hauck and R. McClarren. A collision-based hybrid method for time dependent, linear, kinetic transport equations. *to be published.*

[14] F. W. Brinkley K.D. Lathrop and Rood P. Theory and use of the spherical harmonics, first collision source, and variable weight versions of the twotran transport program. *Technical Report LA-4600*, 1972.

[15] R. G. McClarren and T. J. Urbatsch. A modified implicit monte carlo method for time-dependent radiative transfer with adaptive material coupling. *J. Comp. Phys.*, 228:5669–5686, 2009.

[16] D. Mihalas and B. Weibel-Mihalas. *Foundation of Radiation Hydrodynamics.* Oxford University Press, 1984.

[17] G.C. Pomraning. The non-equilibrium marshak wave problem. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 21:249–261, 1979.

[18] G.C. Pomraning. Radiation hydrodynamics. *Technical Report LA-UR-82-2625*, 1982.

[19] Arthur Forster Guy Ledanois Raymond Alcouffe, Robert Dautray and B. Mercier. *In Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics*, volume volume 240 of Lecture Notes in Physics. Springer Berlin / Heidelberg, Florida, 1985.

[20] O'Dell R.D R.E Alcouffe and F. W. Brinkley Jr. A first collision source method that satisfies discrete sn transport balance. *Technical Report LA-UR-89-304*, 1989.

[21] G.C. Pomraning R.H. Szilard. Numerical transport and diffusion methods in radiative transfer. *Nucl. Sci. Eng.*, 112:256, 1992.

[22] Thomas A. Brunner Ryan G. McClarren, James Paul Holloway. Analytic solutions for time-dependent, thermal radiative transfer in several geometries. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 109:389–403, 2008.

[23] T. J. Urbatsch and T. M. Evans. *Milagro Version 2, An Implicit Monte Carlo Code for Thermal Radiative Transfer: Capabilities, Development, and Usage*. Los Alamos National Lab La-14195-MS, 2005.

Accelerating a Metropolis Random Walk and Immersion-Method Saddle-Point Algorithms

(Peter Moller, mentor)

Accelerating a Metropolis random walk and immersion-method saddle-point algorithms in multidimensional nuclear potential-energy spaces

Justin Willmert[†]
School of Physics and Astronomy
University of Minnesota
Minneapolis, MN
willmert@physics.umn.edu

Kemper Talley[†]
Bredesen Center for Interdisciplinary
Research and Graduate Education
University of Tennesse–Knoxville
Knoxville, TN
ktalley5@utk.edu

under the mentorship of

Peter Möller
Nuclear & Particle Physics, Astrophysics & Cosmology, Group T-2
Los Alamos National Laboratory
Los Alamos, NM
moller@lanl.gov

as participants of

Los Alamos National Laboratory's[*]
Computational Physics Student Summer Workshop
Sponsored by LANL's Advanced Scientific Computing Program

Abstract

In this paper we present an analysis of and improvements on the Möller et al. [8] utility used to simulate the fission-fragment charge yield distributions by the Metropolis method. Due to particular energy-surface properties, particularly as mass decreases, the random-walk procedure could wander for very long tracks and required prohibitively long run times to collect sufficient statistics. Our work has decreased the execution time by taking advantage of parallelization available to symmetric multiprocessing (SMP) systems exposed through the standard OpenMP language extensions and libraries [13]. Because of the intrinsic parallelizability of the random walk, our efforts have demonstrated that nearly linear speedups with the number of processors are possible within certain critical sections of the algorithm. The same code modernization and application of OpenMP constructs has consequently also resulted in a greatly accelerated minima search and associated saddle-point determination. Important for future work, the parallelization effort has resulted in modular code which can more easily be reused than the original monolithic program.

[*]Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

[†]Also Nuclear & Particle Physics, Astrophysics & Cosmology, Group T-2

1 Introduction

High-performance computing has permitted many advances in all branches of physics including the theoretical work being done in nuclear physics. In the past, relatively simple models existed for calculating the energy of a ground-state nucleus by using the liquid-drop model. As computing capabilities have increased, though, so has the complexity—and maybe more importantly the associated computational costs—of the models. One such model is the macroscopic-microscopic model used to calculate nuclear parameters for a wide variety of shapes. Comparing the charge/mass distribution and associated asymmetries, such as in the particular case of ^{188}Hg, proves the efficacy of this model and therefore the utility of such computations. An important goal beyond the theoretical work to generate these models is then to implement them in computationally efficient algorithms.

2 Mass models

A naïve attempt at calculating the masses of nuclei would simply sum the total masses of all protons neutrons, and while this will provide a reasonable first estimate, detailed predictions are sensitive to the inaccuracies inherent in this method. The issue stems from Einstein's energy-mass relation wherein the (negative) binding energy, which can be both theoretically and computationally difficult, has important consequences on the mass and on other important properties of the nucleus. An improved technique might attempt a full quantum-mechanical calculation of the mass, but because the nature of the strong force makes the requisite quantum chromodynamics calculations computationally impractical if not impossible, mass models are of critical importance to the study of nuclear interactions. Since masses can experimentally be measured with high precision, this allows mass to be a valuable instrument for testing various models. In the following sections, we review several important milestones in the mass models that ultimately lead to the specific model used in our calculations, the macroscopic-microscopic model (§2.5).

2.1 Liquid drop model

One of the first successful mass models utilized was the liquid-drop model (LDM). It was developed before the discovery and description of the strong force, and therefore it is a purely phenomenological model constructed from the patterns seen in experimental data [2]. Experiments show that the nucleon density is nearly constant, proportional to the nucleon number A. This naturally led to a description of nucleons as an incompressible fluid with associated volume (A) and surface ($A^{2/3}$) interaction terms. The volume term is the attractive term due to the mutual binding between nucleons while the surface term is a repulsive correction term because the surface nucleons have fewer neighbors than the interior nucleons. Finally, protons repel one another due to the Coulomb interaction, which is repulsive and gives rise to a term proportional to $Z^2 \cdot A^{-1/3}$.

Further corrections to the classical model can be included from quantum mechanics. An asymmetry term accounts for the energy balance involving the neutron excess, and the particular form for the term shown below (coefficient a_A) can be derived from considering the nucleus to be a two-fluid Fermi gas and keeping the first-order expansion term of the kinetic energy [10]. (In fact, the zeroth-order expansion term reproduces the volume term, confirming it as an appropriate choice.) The final term is the pairing term ($\delta \cdot A^{-1/2}$) which accounts for the observation that nuclei have increased stability when they have even numbers of protons and/or neutrons, interpreted as having complete spin pairs. These two additional quantum terms together with the classical terms produce the Bethe-Weizäcker or semi-empirical mass

formula:

$$M(A,Z) = Zm_P + (A-Z)\,m_N - \frac{E_B}{c^2} \tag{1}$$

where

$$E_B = a_V A + a_S A^{2/3} + a_C \frac{Z^2}{A^{1/3}} + a_A \frac{(A-2Z)^2}{4A} + \frac{\delta}{A^{1/2}} \tag{2}$$

The coefficients (a_V, a_S, a_C, and a_A) are fitted constants over a range of nuclei while δ is fit separately for nuclei with even-even, even-odd, or odd-odd proton/neutron numbers.

Both the LDM and semi-empirical mass models as presented assume spherical (typically ground state) nuclei. For excited nuclei—and of particular interest, nuclei undergoing fission—the nucleus can be deformed. Appropriate modification of the classical terms in Eq. 2 can account for an ellipsoidal nucleus; we instead discuss shape corrections in the context of the macroscopic-microscopic model in §2.3, but further discussion of the Weizäcker formula can be found in Povh et al. [10, §3.3] or Das and Ferbel [2, §5.2.1].

2.2 Shell model

In contrast to the LDM macroscopic interpretation, shell models use a microscopic view as that of independent particles moving within a nuclear potential, and the associated single-particle orbitals are solved quantum mechanically. Because the nucleus is not a dense system—the ratio of the closest packed volume of the nucleus to the actual volume is ≈1/100—nucleon-nucleon forces deviate very little from a mean effective potential [11, §2.1]. Thus the nucleus can be considered as made up of these independent particle states and solved in a manner analogous to the idea of the Hartree-Fock potential of the atom. Also like the electron in the atom, the single-particle solutions have degenerate states that lead to distinct shells. These shells (the magic numbers) serve as the basis for further corrections to the binding energy, particularly in the Strutinsky method, in a manner parallel to the use of the noble gases for calculating ionization energies. The shell model theory is extensive, and as such, we refer the reader to Ring and Schuck [11] or Wong [16] for a more complete discussion. We instead move on to introduce the deformation which is instrumental in constructing the introductory macroscopic-microscopic model.

2.3 Deformation

Light nuclei are almost always spherical due to the nuclear force, but massive nuclei have a more detailed energy balance. The Coulomb force grows more powerful as Z increases, and since Coulomb forces are inversely proportional to the square of the distance, the most bound states result by putting protons as far away from one another as possible. Because nuclear volume is invariant, though, the shape must change. Adding the saturation property which leads to further decreases in binding energy per nucleon for higher A, the interplay between these forces may favor a non-spherical shape.

In particular, nuclei are spherical at closed shells (magic numbers), but as nuclei progress from one major shell to another they transition through prolate shapes (elongated along the z-axis) and then oblate shapes (flattened at the poles) as they approach the next major shell. The reason for this preference in shape is discussed in detail in Wong [16, Ch. 6]. The shape of the nuclei can be reasonably expected to change the distribution of the individual nucleons and as such will change the overall binding energy. Each shape corresponds to a distribution of nucleons that influence the nuclear binding energy considered in the calculations of the macroscopic-microscopic model (§2.5).

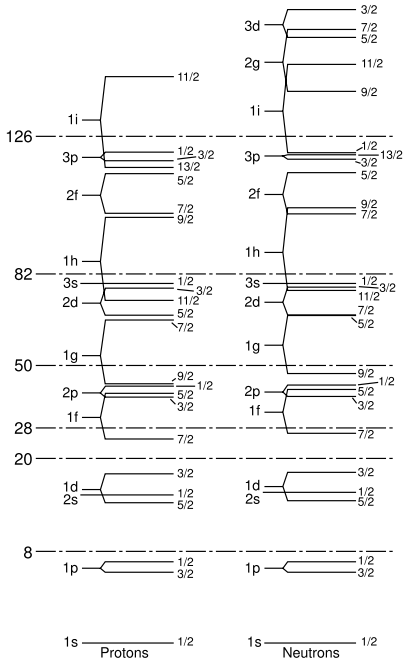


Figure 1: A schematic showing the nuclear levels in the shell model for protons and neutrons with the spin-orbit coupling included. Note the occurrence of magic numbers which correspond to particularly stable configurations. From Povh et al. [10].

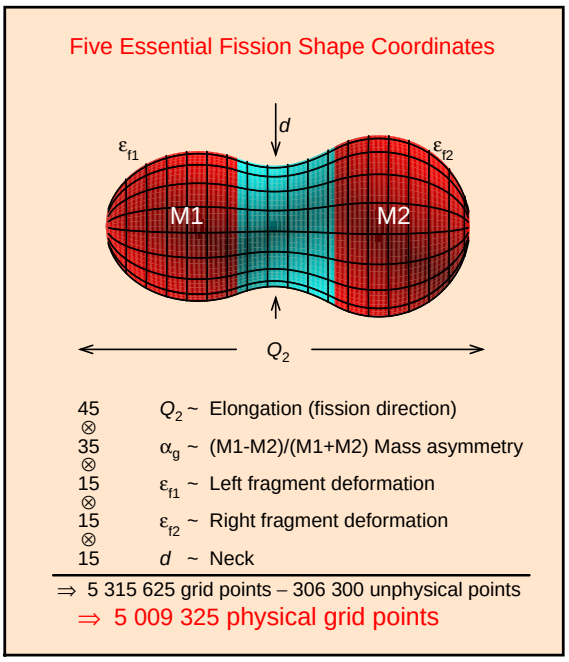


Figure 2: In the Möller et al. [8] presentation, five physical quantities parameterize the shape and energy of the nuclei. Each quantity can be related to the macroscopic-microscopic parameters typically found in the literature. Note that there are 306,400 unphysical points in the model that are included strictly for computational purposes. From Möller et al. [8].

2.4 Strutinsky method

Neither the liquid-drop model nor the shell model are independently capable of accurately calculating the binding and mass energy of all nuclei. The liquid-drop model captures the bulk trend but cannot accurately describe individual nuclei. In contrast, the shell model can calculate the properties of magic nuclei well, but fails to predict the bulk properties. In order to span the gap between both models, Strutinsky created a shell correction procedure which retains the working qualities of both models [12]. It reproduces the experimental ground-state energies as well as their dependence on deformation parameters.

The Strutinksy method deals with the fact that the nuclear binding energy has oscillations from the LDM model due to shell closures. The oscillatory nature of the binding energy between shells has its maximum at the shell closures (i.e. magic numbers: 2, 8, 20, 50 ...). In the shell model one would calculate the energy by including this oscillatory part plus an "average" level density, however, this average was wrong. Strutinsky instead replaced it with the LDM energy while still keeping the fluctuating part from the shell model. More details of how the oscillatory energy can be extracted from the shell model are found in Strutinsky's paper [12].

2.5 Macroscopic-microscopic model

The macroscopic-microscopic model uses smooth trends from a macroscopic model and local variations from a microscopic model in order to calculate global properties. The total potential energy—dependent upon Z, N, and various shape parameters—is the sum of these macroscopic and microscopic terms.

$$E_{\text{pot}}(Z, N, \text{shape}) = E_{\text{macro}}(Z, N, \text{shape}) + E_{\text{micro}}(Z, N, \text{shape}) \tag{3}$$

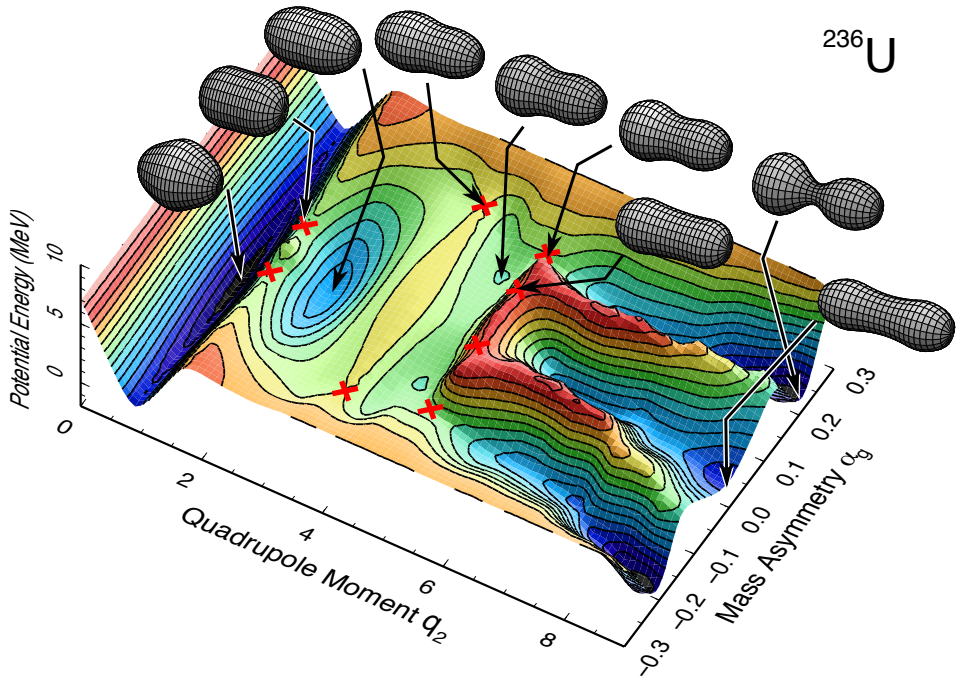


Figure 3: Two-dimensional surface projection of the five-dimensional potential-energy surface used. Key features such as minima, saddle-points (red crosses), ridges, and valleys have been captured in this projection. From Ichikawa et al. [6].

For a specific nucleus, the macroscopic term corresponds to the LDM energy, and the microscopic term is determined by the following prescription. (1) A shape is prescribed using all five parameters given in Fig 2. (2) A single-particle potential with this shape is generated, including the spin-orbit term. (3) The Schrödinger equation is solved for this deformed potential, generating the corresponding single-potential levels and wave functions. (4) The shell correction is calculated by use of the Strutinsky method. (5) Finally, the pairing correction is calculated in the BCS or Lipkin-Nogami method. By this method, the energy hypersurface can be computed for each of the 5 million shapes. A full description of the calculation can be found in Möller et al. [8, 9].

This energy surface—such as is shown in Fig. 3 in a two-dimensional projection of the full five-dimensional surface—is then used as the base for the Metropolis method (discussed in §4) that calculates various properties about the fission of a selected nuclei. This energy surface can be used to search for minima and saddle points as well, both important features that are used to compute further properties of nuclear fission.

3 Saddles in multiple dimensions

Barrier energies provide powerful criterion on transitions in many physical systems, including nuclear deformations. It is important, then, to calculate the barrier energy from the data contained within the energy hypersurface. Computationally this corresponds to finding the saddle-point between a given pair of minima or a minimum and the fissioned (or scissioned) state. These saddle-points correspond to the critical deformation of the nucleus at which the nucleus will be irreversibly committed to fission. It is important to note that there are many saddle-points on such a hypersurface. However, the points of interest are those that correspond to barriers. Möller et al. [8] discuss at length how the typical path-minimization methods for determining saddles between minima in multiple dimensions is insufficient and can produce incorrect or misleading results. We have assumed the use of the immersion method which

Möller et al. argue is the correct strategy, but an interested reader can refer to their paper for a discussion on the pitfalls of the path-minimization methods. We first begin by introducing the immersion method in a concrete, physical example and describe the Möller et al. implementation specifically in §5.2.1.

3.1 Immersion method

The immersion method for determining saddle-points is described in an easily accessible way by Hayes [5]. He presents the method in the context of solving the continental divide problem, namely, how does one determine the location of the continental divide? In a single-dimensional landscape, the problem is trivial with the divide defined by the maximum value along the axis. Already in two dimensions, though, the maximum on the height gives a clearly incorrect answer; in an example using North America as the prototype for a two-dimensional surface, Hayes gives the example that "when you search out the highest point in the lower 48 states, you find yourself atop Mount Whitney, in California, elevation 4,418 meters. But Mount Whitney is nowhere near the continental divide." A second method Hayes considered was to classify each point by the features of its 4 cardinal neighbors—whether a neighbor is higher or lower—and follow a continuous ridge across the continent. He concluded, though, that the 81 combinations in 16 classes of configurations was an untenable idea. In our case, the situation is drastically worse. Having 242 nearest neighbors in five dimensions leads to an incredible $2.9 \cdot 10^{115}$ configurations! Considering the problem further, he concluded that purely local methods are insufficient to determine an intrinsically non-local feature.

The solution as Hayes and others—particularly in geography and topography—have made use of (see Vincent and Soille [15]) is to instead perform a global operation and raise the sea level. The continental divide is then simply the line where the Pacific and Atlantic Oceans' waters meet without mixing. Digitally this can be achieved by keeping track of which locations are dry and which are wet, incrementally increasing the water level on both sides at equal rates.

4 Metropolis random walk

The Metropolis random walk is a Markov chain Monte Carlo method for producing a random walk through the potential-energy hypersurface. A pictorial analogy is that of a drunk man walking around with his path more favorably choosing some directions over others due to topographical features such as hills or valleys. The drunk man is more likely to move down the hill than up it, but there is still a chance that he will climb it. In a similar manner, we simulate the shape evolution of a nucleus by having it "walk" through the potential-energy hypersurface. We initialize the algorithm by starting the walk at a minimum with a given excitation energy above the surface, and evolve the shape according to a simple algorithm until scission occurs: (1) Choose a neighboring cell at random, corresponding to an incremental shape change. (2) If the new shape has a lower energy than the current shape, immediately walk to the cell. (3) Otherwise, step to the new cell only with a probability equal to a Boltzmann-like factor from the energy difference and an effective temperature. (4) Check whether the neck radius is compatible with a scissioned state. If so, the algorithm is finished and the final shape parameters are recorded, otherwise the process repeats for another step in the Metropolis random-walk path.

A single Metropolis random walk is insufficient to provide insights into the dynamics, so an ensemble of paths are collected. From the ensemble of results, various properties such as the mass/charge distributions of the decay of nuclei can be predicted. Confidence in the conclusions drawn requires sufficiently large statistics, and because the procedure must be run for thousands of nuclei, producing a computationally efficient implementation of the Metropolis random walk is important to active research goals.

5 Implementation

We have made an attempt to follow Donald Knuth's advice as stated in his popular quotation, "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil" [7]. Our improvement efforts were directed by the results that were collected from various performance analyzers. We did this for two primary reasons. First, the time we had to complete our work was limited, so we could not afford to waste our time with "optimizations" which later proved to be ineffective. Second, highly optimized code often looks very different than its unoptimized form, usually at the expense of readability. Just for the sake of maintaining easily understandable and maintainable code it is important to avoid unnecessarily obfuscating the code.

For example, an early preliminary idea of what caused the original algorithm implementations to perform poorly was poor cache use. The common solution to such a problem is to change the data storage model to be more cache-friendly. These data structures, though, are often very complex and using them directly in algorithm implementations is nontrivial and error prone. In a language like C++, powerful template metaprogramming techniques can be used to abstract the data structure so that a simple interface is presented while simultaneously mitigating the run-time cost through compile-time optimizations. Fortran does not have the capacity for the same type of compile-time optimized metaprogramming, so we would have had to choose between using an error-prone direct implementation (thereby increasing the debugging time) or sacrificing some performance to build and use an abstract interface. Furthermore, Fortran's lack of an extensive standard libraries (like C++'s STL or Boost) on top of which to build more complex data structures and algorithms would have required writing the basic building blocks from scratch, greatly increasing the implementation time.

Our first goal was to transform the original serial implementation of the random walk into one that could run individual walks in parallel, but before parallelization constructs could be used, the existing code needed to be updated and cleaned. The reasons will be discussed in §5.1. In the process of updating the random-walk code, several reusable sections of code made the reimplementation of the saddle-point algorithm convenient, and several algorithmic changes we introduced will be discussed in §5.2. Throughout this process we made use of two performance analysis tools: the GNU Project's `gprof` [4] and Valgrind's `cachegrind` tool [14]. Because the outputs from both programs tend to be long, we do not present the results and analysis inline and have instead placed the discussions in Appendices A.1 and A.2. We also explored the use of the Tuning and Analysis Utilities (`TAU`) [3] which makes use of hardware counters available on many modern architectures. Configuration problems and associated complications made this method of analysis ineffectual within our time table and did not factor into our analysis. As a consequence, `TAU` will not be further discussed here, but it presents a promising task for a future project to expand upon the analysis we have already performed.

5.1 Updating the code base

The first stage in implementing parallelization features into the random walk and saddle determination algorithms was to update the code base from FORTRAN77 to Fortran 95. The original FORTRAN77 code was not structured in a way that allowed for the use of OpenMP [13], our choice of parallelization technology. Another motivation for the modernization was to create a self-consistent software package. We were provided the complete FORTRAN77 source for random walk routines from Peter Möller and code snippets for reusable routines such as routines to load the data from disk, but we were independently implementing the saddle-point algorithms from scratch. To make best use of our time, we preferred a consistent coding style and set language features.

The first stage of changes focused solely on updating the syntax from FORTRAN77 style to Fortran 95. These changes included moving away from the fixed column layout to free form so that logical

code blocks could follow a consistent indentation style, replacing use of labeled `CONTINUE` statements with equivalent `end do` or `end if` block constructs, and converting extensive use of `GOTO` statements with `if/else` pairs. An ongoing effort began at this stage as well to explicitly declare all variables rather than using implicit typing. Requiring explicit variable declarations aided development by enabling compile-time warning and error messages that were capable of catching more type mismatches or unused variables, greatly helping the cleanup effort. Because the code had to be disentangled, though, often the `implicit none` feature could not be enabled for a procedure or module until several more stages of our code updates had also been completed.

The next stage of development was to extract the monolithic code base into independent modules. There are several procedures which were reused in every application, but the original implementation used a copy-paste methodology for providing the necessary routines. Modules also have the advantage that an application programming interface (API) can be generated which declares public interfaces and hides routines intended for internal use only. The utility of this strategy is demonstrated by example of the routines used to load the matrix data from disk. The data is stored as ASCII (plain text) values, and the original loading routine provided an interface to read the data and arrange it into the five dimensional matrix that is used in subsequent calculations. Parsing the ASCII, though, is a slow process; binary files can be loaded into memory in a fraction of the time, but they also pose a portability problem (data alignment, endianness, etc). We introduced two new specialized methods to load either the original ASCII data which is guaranteed to be portable or load a binary dump of the in-memory matrix that may be compiler, architecture, or Fortran version dependent. The original interface was then used to automatically load the binary cache of the data if it existed or read the ASCII file and generate a cache file for the next use. In this way, all programs compiled against the module automatically gained a new feature without requiring any additional work.

The final major stage of code cleanup was to replace extensive use of `COMMON` blocks to share data between procedures with explicit passing in procedure calls with user-defined derived types. In addition to improving code readability by making possible side-effects more obvious, the step was also critical to adding OpenMP support. `COMMON` block require that the OpenMP scope (such as `threadprivate` or `shared`) be declared at every usage. This is error-prone since any missing update could lead to both compile-time and run-time errors. With derived types, though, a variable must only declare its scope once where the parallelization instructions are used. Subsequent use of the data in calling child procedures automatically make use of the correct data, whether that be a local thread copy or a shared global resource.

5.1.1 Parallelization with OpenMP

The large effort necessary to update or rewrite code was rewarded through the simplicity with which parallelization was added using the OpenMP framework. In an ideal situation, parallelizing a loop without OpenMP directives would effect no extra changes to the code. In more realistic situations, though, some shared state typically needs careful consideration and possibly specialized code. Keeping special cases to a minimum, though, helps ensure that the program output and behavior is identical whether run serially or in parallel on an SMP system. In Fortran, OpenMP declarations are added as comment lines with a special syntax that instructs an OpenMP-aware compiler while simultaneously maintaining compatibility with OpenMP-ignorant compilers.

We added `!$OMP PARALLEL DO` statements surrounding two so-called hot loops—critical loops that consume the majority of computational resources: one in the random walk algorithm and one in the virtual flooding used to identify saddle points. In the random-walk algorithm, we parallelized over multiple paths. The act of a random walk is completely dependent on its history so an individual path cannot split into multiple work units. We were interested in generating an ensemble of paths, though,

and since every path is inherently independent from any other, it is almost a prototype for parallelizable algorithms; only a small amount of end-state data must be collated across all paths. In the flooding algorithm, a complete and coherent flood map is required prior to expanding the flood, so parallelizing must be restricted to a single step, where a single step is defined as growing the flood to only its nearest neighbors (subject to energy constraints). Utilizing the popular idea of double-buffering from computer graphics, though, each individual step can be parallelized across the examination of the greater than five million elements in the flood map and energy matrices. Given an input flood map, a parallelized flooding step can iterate across all cells in the matrix, examining each cell's nearest neighbors in the input map, and outputting any changes to a different output map. This guarantees that all threads see the same input state no matter in which order the map is updated.

A few modifications required to have the OpenMP parallelization function properly are of note: (1) large stack-allocated variables must be avoided, and (2) we swapped out the original random number generator for that used in the Monte Carlo N-Particle Transport Code version 5 (MCNP5) software package [1].

Stack variables pose problems for OpenMP parallelized code because the program stack can quickly be exhausted, leading to segmentation-fault errors. This is because every instance of the procedure running in parallel requires its own copy of stack variables, so the memory requirements scale with the number of processors. The simple fix for this is to make use of dynamically allocated memory that is carved out of the heap. The heap is typically much larger than the stack, and can grow dynamically even larger if necessary. The downside is that extra effort is required to release the memory appropriately to avoid leaks that consume extra resources.

We switched to the MCNP5 random number generator because the original generator was incompatible with running in parallel. First, time would have been required to properly protect common blocks and global variables for use in parallel code. Doing so would have been possible, but without a skip-ahead procedure, the effort would have been wasted. A critical goal of parallel algorithms is to produce identical results no matter how many threads are used during execution. In an algorithm that makes use of random number sequences, this means that a particular identifiable part of the problem should start from a definite location in the pseudo-random number sequence. The original generator could only accomplish a skip ahead by generating every value in the intermediate sequence, a very costly process. In contrast, the MCNP5 generator, because it was designed to be used in a parallel code, was created with an efficient skip ahead procedure; only a few algebraic operations are required to skip to any arbitrary location in the sequence. We made use of this feature to produce identical results whether our code was run on a single processor or sixteen.

In addition to adding parallelization to the flooding process, several algorithmic changes were implemented. These changes will be discussed in the following section before we return in §6 to present the run-time performance gains we measured.

5.2 Algorithmic updates

As part of an introductory exercise to the code base and methods, our mentor tasked us with implementing our own version of the minima identification and saddle-point determination algorithms. Fruitfully, this effort provided more than just an educational exercise: it also resulted in several enhancements to the algorithm over the original implementation. We were provided high-level, abstract descriptions of the algorithm, but in the absence of a reference code, our approach and implementation choices were different. In the next few sections, we outline these differences and describe how they have contributed to our more efficient solution.

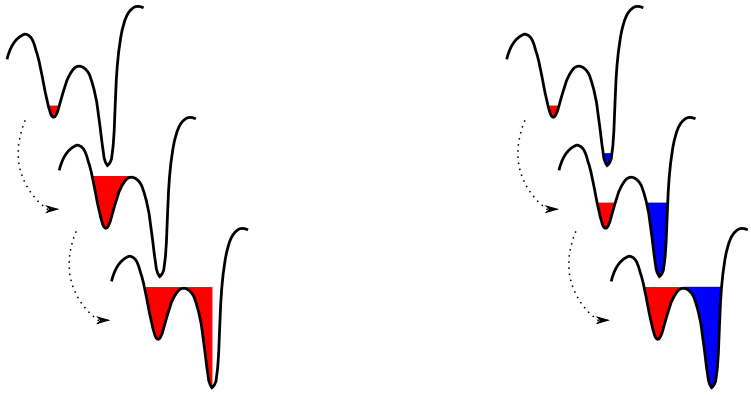


Figure 4: A simplified comparison of the original and new flooding algorithms implemented for saddle determination. (a) In the single-flooding algorithm, only one of the minima are made wet. The flood is stepped successively higher until it floods over the top of the saddle point, and then it must flow down until the second minimum is detected as having been made wet. (b) For simultaneous flooding, both minima are initialized as wet, but with two different, distinguishable floods. The lower minimum's flood is raised individually until at the same level with the higher, and then the process progresses such that both floods are always at the same elevation. Both are raised until the floods collide at the saddle point.

5.2.1 Original Möller et al. implementation

Möller et al. search for the saddle point between two minima labeled the entry and exit points. An auxiliary matrix is created which associates a property of "wetness" with each cell in the original energy matrix. This auxiliary matrix—which we refer to as the flood map—tracks the virtual flood used in the immersion method. They begin by setting only the entry point to be wet in an otherwise completely dry flood map. The flood is expanded by repeatedly processing every cell in the flood map. A particular cell is made wet if it simultaneously has a corresponding energy less than a given threshold and has a neighbor which is already wet. They choose the initial threshold to be 1 MeV above the entry point's energy. The flood is expanded until either an iteration where no changes are made in the flood map occurs—signaling that the flood has expanded as far as is permitted by the energy surface and the threshold limit—or the exit point becomes wet. In the former case, the exit point is examined to determine whether it is wet. If it is not then the threshold was lower than the saddle energy between the entry and exit points, and the threshold can define a lower bound on the saddle energy. The threshold is then incremented by another 1 MeV, and the process is repeated. If instead the exit point is wet, then the threshold is an upper bound on the saddle energy. Together the lower and upper bounds define an interval within which the saddle energy must exist. The saddle energy can be defined to a desired precision by continually decreasing the increment size from the initial 1 MeV by a factor of 10 when necessary.

5.2.2 Simultaneous flooding

One of the first optimizations we made was to decrease the number of times that the entire energy matrix had to be traversed by flooding both minima simultaneously. Möller et al. mention the possibility of doing this by making use of two flood maps—one for each flood—but dismiss the implementation as being needlessly complex. Rather than making use of two flood maps, though, we were able to achieve simultaneous flooding with only a single flood map by simply identifying each flood with a different unique integer and non-flooded cells with the zero value. In theory, this method can be used to simultaneously flood as many minima as unique values can be stored in the map (255 for the one byte cells currently used). Simultaneous flooding reduces the number of iterations required by avoiding unnecessary flooding. With over 5 million cells in the matrix, shaving off iterations can rapidly save

computational time. Möller et al.'s method iterates until the flood map is left unchanged after a pass over the entire matrix. They are then able to check when the the bound was too high or too low based on the exit point's "wetness". Our implementation, though, is capable of detecting collisions between the simultaneously expanding floods, possibly long before all eligible points have been flooded. If more than one collision occurs, then we already know the threshold is too high and can restart the search with a tighter bound. Additionally, we gain the ability to automatically know when the process has reached sufficient convergence by noting when only a single collision is generated during a single pass.

5.2.3 Saddle convergence

Another small optimization came from a different choice in the convergence technique used to identify the saddle energy. As described in §5.2.1, the Möller et al. implementation converged on the energy by using a linearly increasing threshold with step sizes that decreased by an order of magnitude when the algorithm detected that the energy had been overestimated. In our implementation, we instead use a binary-search style convergence algorithm. The lower bound is set by the minima energies, but there is no method a priori to know what the upper bound should be. We determine this upper bound by stepping the threshold by 1 MeV steps much like the Möller algorithm until the first overestimate is detected. After both the lower and upper bounds have been determined, though, the process proceeds by a binary search. The threshold is set to the midpoint between the upper and lower bounds. If the threshold is still an overestimate, the upper bound is decreased to the threshold, and the process is repeated. If instead the flood reaches the threshold and an entire pass results in no changes to the flood map, then the lower bound is increased to the threshold, and the process is again repeated. This process of bisection and flooding is continued until a single collision point is detected which indicates that the threshold is a good estimate for the true saddle energy.

The gains measured, though, were modest at best. Raw timings of the time to complete a single saddle determination were lower for the binary search case than when reprogrammed to make use of a linear search like Möller et al.'s original implementation. In addition, we identified using `gprof` (see §A.1 for more information) that a particularly expensive operation—creation of a backup flood map for use when the upper bound is too high and the process needs to restart from a state before any collisions have occurred—requires a smaller fraction of the total run time in the binary search case. On a particular test case, the relative weights decreased from 0.20% for the linear search to 0.03% for the binary search. The significance to overall run time is very low, though, so any future performance gains are unlikely from exploring this particular implementation detail further.

5.2.4 Bounded-box flooding

The reason the flooding operation can be so computationally expensive is because the algorithm potentially examines the more than five million cells over many complete iterations. The motivation for the bounded box addition was to decrease the number of elements which must be examined on each element by excluding regions which are known to be completely dry and cannot become wet in the next iteration. Figure 5 demonstrates the method in a simplified two-dimensional case, and the method is directly expanded to the five dimensions contained in our problem. When the two flood points are initialized at each respective minimum point, the maximum and minimum extents are remembered in each dimension. The bounding box is then defined by subtracting 1 from the minimum values and adding 1 to the maximum values. Since the flood can only grow by a single cell into a neighbor during a single iteration, this is sufficient to define the next maximum possible extents of the flood. The flooding algorithm must then only loop over cells contained within the bounds of the hypercube during a given iteration. If during the loop the flood is extended to reach a face of the bounding hypercube, the extent in that direction is

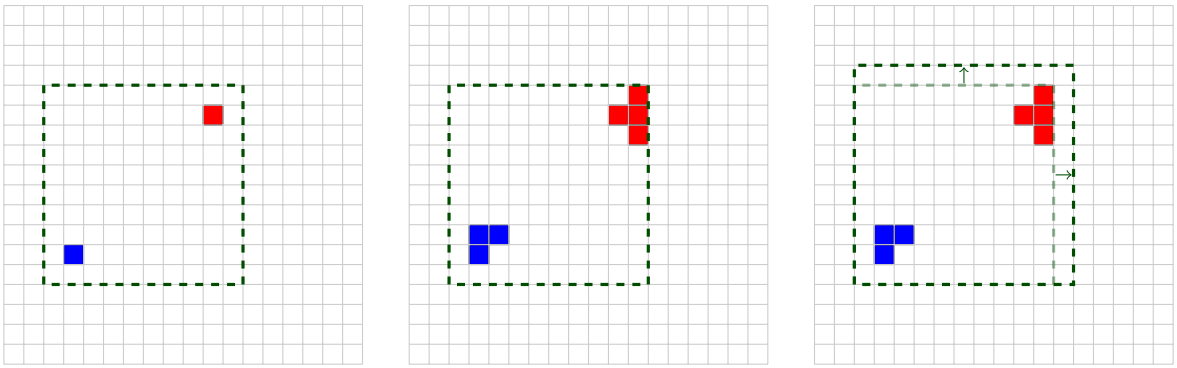


Figure 5: The three major stages in the bounding-box optimization to the immersion method for a simplified 2D example. (a) The flood map is initialized with an appropriate bounding box. The matrix traversal starts at the top-left corner *of the bounding box* and proceeds in column-major order within established bounds. (b) During traversal, cells with a wet neighbor are also made wet within the constraints of the energy surface values (not depicted). Some expansions may cause a flood to expand into the boundary such as at the top right. (c) The bounding box is expanded in any dimension's limits to re-establish a single-cell border around the outermost flooded cells. The process restarts as in (a) from the new box's top-left cell.

pushed outward (toward zero for a minimum and toward infinity for a maximum extent) to maintain a hypercube sufficient to contain the *next iteration's* maximum possible size.

The performance gains were again modest, on the order of a few percent or less which is not to be entirely unexpected in our limited tests upon only a couple of energy surfaces. We expect that the the most significant gains will occur at for early steps where the flooded regions are small, especially if the minima are located near each other in the energy surface. At further steps, the flood can spread to span across the extent of some or all dimensions, thereby requiring a traversal of nearly every cell in the matrix and eliminating any performance gains. In particular, the common case of locating the saddle between the ground and scissioned states almost completely spans the elongation parameter, and therefore a large fraction of the entire hypercube is included within the bounds already at initialization. Furthermore, confident conclusions are only reasonable after examining the performance change for a large number of energy surfaces, which is goal of future studies.

6 Performance analysis

The final analysis results shown in the following subsections makes use of timers wrapped around simple stub routines which are capable of running an isolated algorithm. This required special care to ensure that extraneous and uninteresting processes such as loading the energy matrix or auxiliary data is excluded from the timings by performing them before the timing begins. In all presented results, the timing gives the average time required to complete the operation over five invocations, and the standard deviation in times is used to the estimated statistical error. We made use of two types of timings: wall (or real) time and CPU time. The wall time is the physical, world time that elapses between two events, as if you were to watch a clock on the wall and note the time interval. This is in contrast to CPU time which is the sum total of time that the processor(s) were busy in computations. On a single processor machine, the wall and CPU times should correspond almost exactly. For SMP systems, though, CPU time will be greater than the wall time since multiple cores were computing in parallel during the same physical time interval. Under perfect circumstances we'd expect the wall time to be inversely related to the number of processors used to compute the problem while maintaining constant CPU time. Because of various forms of overhead, any

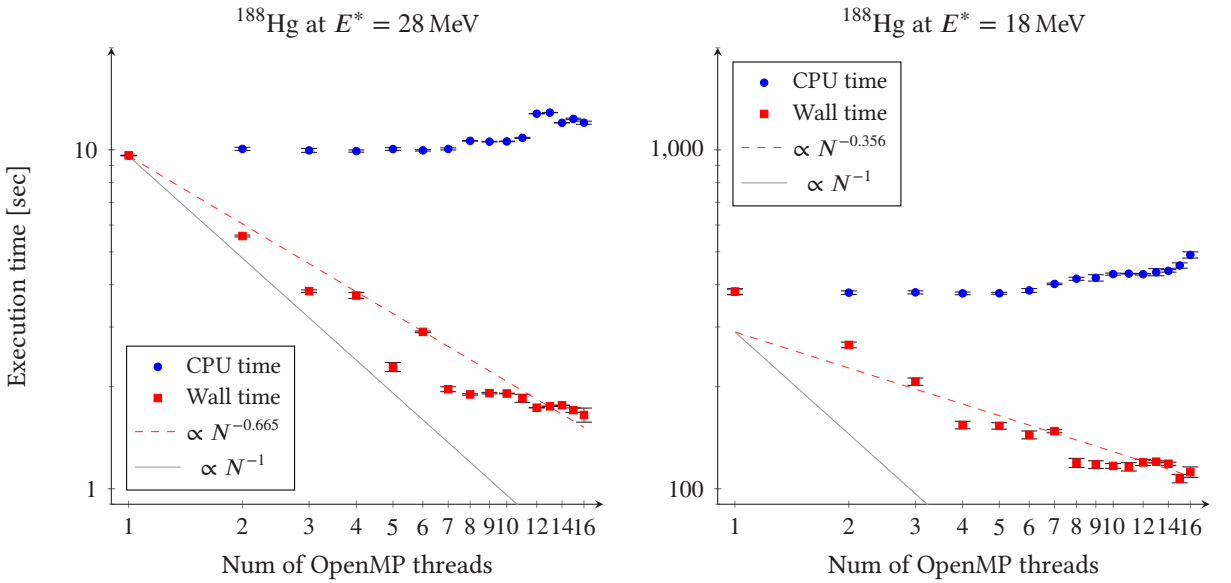


Figure 6: Shown are the run-time performance analyses for 16 random walk paths over the ^{188}Hg energy hypersurface for excitation energies of 18 and 28 MeV. A perfectly parallelized algorithm would be expected to have constant CPU time while decreasing the wall time proportional to the number of OpenMP threads. Taking the lesser performance of the 18 MeV case together with the systematic zig-zag deviations in wall time, long running single paths which hold completion of the process is a significant factor in total completion time.

real-world result will fall short of these expectations. We collected timings for both the random-walk and saddle-search algorithms in 16 timing trials which sequentially incremented the number of processors in use from 1 to 16.

6.1 Random walk performance

The random-walk algorithm was tested using the ^{188}Hg energy hypersurface with excitation energies E^* of 18 MeV and 28 MeV. Each trial was asked to calculate the yield distribution for 16 paths, chosen to correspond to the number of processors available so that at maximum utilization, each processor is assigned a single path. The results are presented in Fig. 6. In both cases, a log-log fit was generated for the wall time and compared against a perfect N^{-1} response.

We see that the wall-time performance as a function of number of threads depends on the problem context strongly; the $E^* = 28$ MeV case, which already required over an order of magnitude less time than the 18 MeV case when executed serially, conforms more closely to the ideal and gains a larger speedup factor. Despite the low convergence rate of $N^{-0.356}$, though, the results are promising since ^{188}Hg represents one of the most difficult energy hypersurfaces for the Metropolis method. Furthermore, the effects of exceptionally long paths are visible in the results. The zig-zag pattern in the timings are not a statistical artifact—the error bars show the standard deviation in timings but are almost invisible on the plots. It is caused by one or two long running paths that continue to run on a single processor after all other paths have finished. Given these considerations, we expect that more well-behaved energy hypersurfaces will achieve more consistent speedups.

Additionally, the CPU time is qualitatively well constrained to a constant; we observe small increases in total CPU time as a function of the number of OpenMP threads, but the rise is not more than would be reasonably expected from overhead inherent to coordinating and synchronizing additional threads.

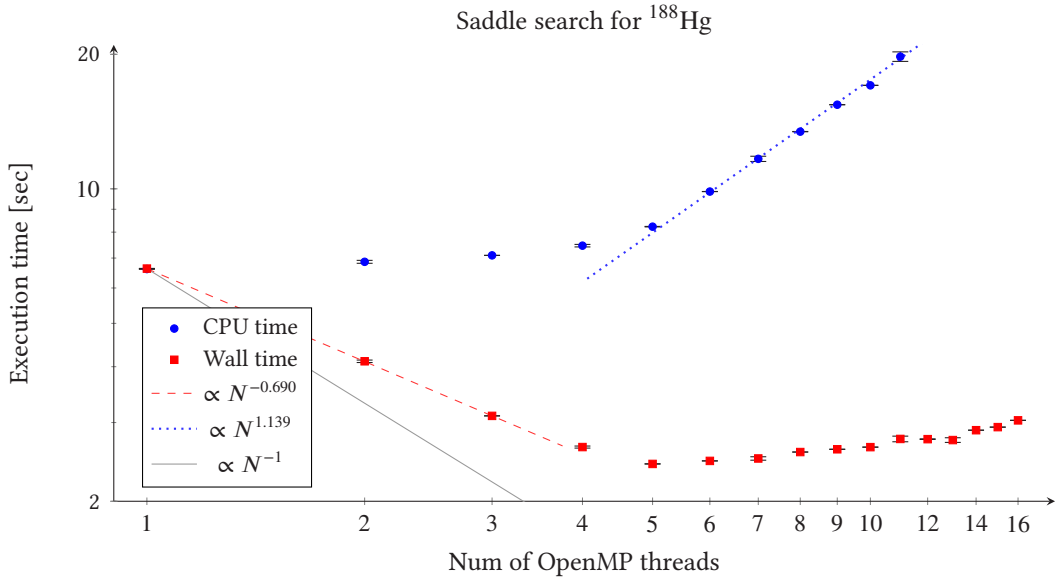


Figure 7: The timing results to find the saddle point between a single pair of minima. The algorithm does not scale beyond approximately 4 processors—simultaneously the wall time hits a plateau while the CPU time increases proportional to the number of processors. A critical bottleneck, likely to be waiting on main memory, currently limits the calculation rate for the saddle-search algorithm.

6.2 Flooding performance

Determining the saddle point for a single pair of minima in ^{188}Hg was also performed to examine the performance characteristics of the flooding algorithms as a function of the number of parallel threads. The results are shown in Fig. 7, and they differ from those of the random-walk tests starkly. First, the convergence proportional to $N^{-0.690}$ only extends to about 4 or 5 processors (the first 4 points were used in the fit) before the performance gains plateau and begin to decrease. In this case, we cannot continue to reduce the real-world computational time by simply throwing a larger number of processing cores at the problem.

In fact as the CPU time shows, we actually waste resources by asking that all 16 processors on our test machine work on the task. Up through roughly the same limit where the wall time decreases, we also see a relatively constant total CPU time. Beyond that limit where the wall-time hits a plateau, though, we see that the CPU time increases dramatically—at a rate approximately proportional to the number of processors in fact! The flooding algorithm makes much more demands on memory, and we suspect that cache misses become significant and each thread begins to wait for data to be retrieved from main memory. Therefore, the most efficient use of computing resources for the saddle-finding procedure is to utilize approximately 4 processors.

7 Future work

As occurs in any project, there are places where there is additional work to be done: (1) Previously mentioned in §6, more reliable and accurate performance analysis can be performed by making use of the `TAU` instrumentation suite to provide access to the hardware counters built into many computer architectures. The sampling and simulation techniques used by `gprof` and `cachegrind`, respectively, proved to be useful tools to do a broad optimization of the algorithms, but hardware counters that provide production, run-time data may give insights into remaining performance bottlenecks. In particular, more

advanced methods may provide further insight into the complex interactions parallelized algorithms can have with the underlying hardware (as suggested by the saddle finding performance in Fig. 7). (2) Extensive performance tests across a wide range of nuclei could provide further insights into bottlenecks or limitations in these updated implementations. (3) Modifications to the saddle-finding algorithm may be possible that would allow the simultaneous identification of saddle points between multiple pairs of minima. The current implementation finds saddle points between pairs serially. (4) Changes to the structure of the energy matrices may lead to further enhancements. For example, eliminating the need for the unphysical points simply for computational purposes could reduce resource requirements. Another option would be to use a different choice of parameterizations which produce computationally favorable energy surfaces; in particular, the flooding algorithms are sensitive to the extents of the flood, so choosing a parameterization which preferentially constrains to a subspace could amplify the effectiveness of the bounding boxes. (5) Alternative storage of the matrices in memory—possibly multiplexing or an as-yet unconsidered format—could provide more efficient cache usage if point 1 can prove where this method would be effective.

8 Acknowledgements

We'd first like to thank in general Los Alamos National Laboratory's Advanced Scientific Computing Program; it is responsible for generously supporting the Computational Physics Student Summer Workshop in which we have participated. A special thank you must go to our mentor Peter Möller for his guidance and in particular his patience as one of us (Justin) was introduced to an unfamiliar field and the other (Kemper) who has much less coding experience. This work would also not have been possible without the use of his code and data as a starting point. Next, a thank you to Scott Runnels for all the organizational work done to make this workshop a success, from the application process and in-processing paperwork to all the lectures provided for our benefit. Lastly, we want to thank all of our peers and other workshop mentors who have provided great sounding boards against which to help refine ideas and strategies.

A Performance profilers

As first mentioned in §5, we made use of the `gprof` [4] and `cachegrind` [14] tools to help guide our implementation strategies. In the following two sections we will introduce each tool, discuss the advantages and pitfalls of each, and provide a short discussion of the conclusions we drew from an example output.

A.1 The GNU profiler

The GNU profiler `gprof` is a sampling profiler. This means it interrupts the execution of the program of interest at periodic intervals in order to record the location of execution within the program. This has several important consequences for how a program is profiled and what conclusions can be drawn from the resultant information.

First, because the profiler samples a running program, the collected profiling data will only reflect the performance for a particular code path. In a complex piece of software such as a web browser, it can be very challenging to ensure that all code paths of interest are executed since the browser may execute very different sections of its code based on its interaction with the user. In our case, though, both the random walk and flooding procedures are simple, single-purpose utilities which do not have alternate modes of usage. This means that we can be confident that our programs are being sufficiently covered to provide representative samples. Further work is required to reach strong conclusions, though—it is known that not all nuclei are equally computationally expensive and make unique demands on resource usage, so a comprehensive review of many nuclei is still needed.

Second, the sampling occurs at periodic intervals which may systematically never overlap with a particular line of code. If we were concerned with corner case performance or micro-optimizations, we would have more reason to worry, but because we were interested in the performance of only the internal hot loops which are executed very many times, we are not concerned with the fact that several iterations may not be counted. The programs still spend an overwhelming fraction of their time working within the hot loops and therefore a fair representative sample should still result.

Additionally, matching executing instructions with specific lines of original code can be inaccurate or misleading. Without any optimizations, the machine instructions can be traced back directly to the code that generated them, but the use of optimizing compilers can make the correspondence less accurate or completely misleading. This is because the optimizers make use of very complex passes which can rewrite the machine instructions in order to make best use of the available resources, but this may also meld the code lines so that the code flow is no longer sequential. A decision must then be made. One choice is to sacrifice program performance (and deviate from the production executable that would actually be executed) to retain accurate code-to-instruction correspondence. The other choice uses the optimized program at the cost of generating profile information which cannot be entirely trusted.

Finally, the sampling method only provides performance data by counting the number of times a particular line of code is executing when the sample is taken. This provides information about where the code is spending its time, but not why. In order to properly optimize a particular line of code, it is necessary to know the reason for the long execution time. For example, the method for reducing the number of cache misses in a line of code will vary greatly from the strategy used to reduce the number of cycles required on the CPU. The output from `gprof` must therefore either be interpreted using heuristics or another tool which can provide extra information, and this is the reason for also making use of `cachegrind`.

Listing 1: A sample of the output generated by the `gprof` analysis tool during execution of the saddle finding utility. The particularly useful data is given in the first and last columns. The first column shows the cumulative fraction of the time spent in executing a particular line of code. The least column then gives the file and line number of the corresponding code.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 75.76     22.54    22.54                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401de4)
  3.90     23.70     1.16                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:68 @ 401b39)
  2.64     24.48     0.79                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:101 @ 401fe4)
  2.61     25.26     0.78                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:71 @ 401cda)
  2.07     25.87     0.62                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:65 @ 401b24)
  1.78     26.41     0.53                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401d7b)
  1.65     26.90     0.49                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:363 @ 408172)
  1.56     27.36     0.47                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401da9)
  1.13     27.70     0.34                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401dbc)
  1.06     28.01     0.32                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401dd1)
  0.99     28.31     0.30                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401dce)
  0.84     28.56     0.25                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401db5)
  0.44     28.69     0.13                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401dcb)
  0.32     28.78     0.10                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:363 @ 40815d)
  0.29     28.87     0.09                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401de0)
  0.27     28.95     0.08                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:63 @ 401b6a)
  0.20     29.01     0.06                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:88 @ 401d24)
  0.18     29.06     0.06                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401d01)
  0.15     29.11     0.05                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:88 @ 401b1d)
  0.13     29.15     0.04                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401fe0)
  0.13     29.19     0.04                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401fc3)
  0.12     29.22     0.04                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:101 @ 401fd9)
  0.12     29.26     0.04                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 408121)
  0.10     29.29     0.03                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:88 @ 401d0f)
  0.10     29.32     0.03                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401d17)
  0.10     29.35     0.03                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401d9a)
  0.10     29.38     0.03                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:101 @ 401fb6)
  0.10     29.41     0.03                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401fbc)
  0.08     29.43     0.03                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:363 @ 408138)
  0.07     29.45     0.02                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401c70)
  0.07     29.47     0.02                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:342 @ 40802e)
  0.07     29.49     0.02                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 408141)
  0.07     29.51     0.02                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:363 @ 408150)
  0.07     29.53     0.02                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 408162)
  0.05     29.55     0.02                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401da2)
  0.03     29.56     0.01                             __moller_flooding_MOD_flood_min_pair (moller_flooding.f95:230 @ 404000)
  0.03     29.57     0.01                             __moller_flooding_MOD_flood_min_pair (moller_flooding.f95:267 @ 4043ff)
  0.03     29.58     0.01                             __moller_flooding_MOD_flood_min_pair (moller_flooding.f95:255 @ 404780)
  0.03     29.59     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 4019ba)
  0.03     29.60     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401abe)
  0.03     29.61     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401d5c)
  0.03     29.62     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401d6e)
  0.03     29.63     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401daf)
  0.03     29.64     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401dc8)
  0.03     29.65     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:94 @ 401f7f)
  0.03     29.66     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:95 @ 401f87)
  0.03     29.67     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401f9b)
  0.03     29.68     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:101 @ 401fd3)
  0.03     29.69     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401fd6)
```

```
  0.03     29.70     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:102 @ 402281)
  0.03     29.71     0.01                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:324 @ 407b63)
  0.03     29.72     0.01                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 4080fa)
  0.03     29.73     0.01                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:339 @ 408441)
  0.03     29.74     0.01                             rang (mcnp_random.f95:180 @ 407b2d)
  0.02     29.74     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:62 @ 401ab2)
  0.02     29.75     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:88 @ 401ab9)
  0.02     29.75     0.01                             __moller_flooding_MOD_flood_step._omp_fn.0 (moller_flooding.f95:91 @ 401db2)
  0.02     29.76     0.01                             __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 408159)
  0.00     29.76     0.00      112     0.00     0.00  __moller_flooding_MOD_flood_step (moller_flooding.f95:38 @ 402740)
  0.00     29.76     0.00       89     0.00     0.00  frame_dummy
  0.00     29.76     0.00        2     0.00     0.00  __moller_flooding_MOD_saddle_point (moller_flooding.f95:287 @ 4076c0)
  0.00     29.76     0.00        1     0.00     0.00  __mcnp_random_MOD_rn_init_problem (mcnp_random.f95:232 @ 406ea0)
  0.00     29.76     0.00        1     0.00     0.00  __moller_datafiles_MOD_load_tensor (moller_datafiles.f95:186 @ 406490)
  0.00     29.76     0.00        1     0.00     0.00  __moller_datafiles_MOD_load_tensor_cache (moller_datafiles.f95:123 @ 405ce0)
  0.00     29.76     0.00        1     0.00     0.00  __moller_flooding_MOD_flood_min_pair (moller_flooding.f95:135 @ 402980)

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

A.1.1 Program preparation & tool execution

Using `gprof` requires additional compile-time tooling to include extra machine instructions necessary for collecting the samples, and therefore `gprof` is only suitable for analyzing programs that can be compiled from source—embedded debug symbols in the program are insufficient. The advantage of the compile-time tooling is that no extra effort is required to generate profiling information; only a single compiler option must be added, and the rest of the work is performed automatically by the compiler. For the GNU Compiler Collection (GCC), the flag `-pg` must be added to compilation. The resultant executable is then profiling-enabled.

Generation of the profiling information is automatic upon execution of the program with the data being written to the file `gmon.out` within current working directory. `gmon.out` is not human readable, though, and must be post-processed to generate the output shown in Listing 1 with a command sequence similar to

```
$ gprof --line flood_test gmon.out > flood_test.gprof-annotate
```

The sequence takes as arguments the path to the binary and profile data (here, both in the current working directory) and emits the human readable form to the standard output (here being redirected to a file). The extra option tells `gprof` that we want line-by-line annotations rather than just function totals.

A.1.2 Profile interpretation

Listing 1 shows the flat profile generated by `gprof`. We have enabled line-by-line counting, so most of the output is for specific lines of code, and function totals only show up near the end of the listing. We were most interested in the lines that account for the greatest proportion of the run time, and `gprof` makes this clear for us, sorting the entries by individual percentage cost by default (the left-most column). In Listing 1, the first line shows that the call at line 91 of `moller_flooding.f95` accounts for over 75% of the run time. Looking up the line in source, this corresponds to a search over neighbors cells in the flooding routine for a flooded neighbor.

With a line identified, we can finally expend the time and effort to explore optimization opportunities. The line is a critical part of the flooding algorithm, so there is not a way a trivial way to reduce the number of times the line is executed. Further, the line is a call to a Fortran intrinsic, so we can safely assume that a hand-written method would be less efficient than the code produced by the compiler and support library maintainers. For this specific case, we identified the largest cost in the program and concluded that we could not optimize this specific line within the time constraints and goals of our research, and therefore moved on to other optimization efforts.

The output from `gprof` was used in a similar manner to help with smaller optimizations already discussed in §5. For example, the linear vs. binary saddle convergence described in §5.2.3 was in part analyzed by noting how the costs changed between `gprof` profile outputs; we identified that the number of calls to a particularly expensive memory call were reduced in the binary search implementation over the linear search.

A.2 Valgrind

In contrast to `gprof`, Valgrind is a simulation profiler which runs an interpreter to track every line as it executes. As a consequence Valgrind will never miss an execution of a line of code, but execution time is significantly increased. The major benefit that makes the cost worth paying, though, is that the simulator is capable of simulating how the code interacts with any software or hardware feature that Valgrind and it's plugins are capable of simulating. For `cachegrind`, this means that we gain the ability to simulate

Listing 2: At runtime, `cachegrind` outputs a summary of the collection information to the terminal in addition to generating an auxiliary data file which can be analyzed later (see Listing 3). The most interesting data occurs in lines 41 and 42 which give the cache miss rate for data-related reads and writes. We see that the rates are less than 1%, initially suggesting cache misses are not a significant effect, but see §6.1 and §6.2 for a more complete discussion.

```
1  ==18642== Cachegrind, a cache and branch-prediction profiler
2  ==18642== Copyright (C) 2002-2010, and GNU GPL'd, by Nicholas Nethercote et al.
3  ==18642== Using Valgrind-3.6.0 and LibVEX; rerun with -h for copyright info
4  ==18642== Command: ./randwalk
5  ==18642==
6  --18642-- warning: Unknown Intel cache config value (0x76), ignoring
7  --18642-- warning: Unknown Intel cache config value (0xff), ignoring
8  --18642-- warning: L2 cache not installed, ignore LL results.
9  Please specify the nuclear parameters
10 Z:
11 A:
12  Enter a new value, or leave blank to accept the default.
13
14 Number of iterations to perform [ 50000 ]:
15 Set the excitation energy [ 6.540 ]:
16 Set the shell correction damping factor [ 60.000 ]:
17
18 Excitation energy (wrt ground) :  6.540 MeV.
19 Absolute ground state energy   : -2.110 MeV.
20 Absolute excitation energy     :  4.430 MeV.
21 EDAMP                          : 60.000 MeV.
22
23  All manual input has been entered. Please be patient as the random
24  walks are performed. Note that if no cache exists, reading in the
25  data files may take some time.
26
27
28 Total rnd count :              8062781
29  Mass normalization check:      199.999985
30  Charge normalization check:    200.000000
31 ==18642==
32 ==18642== I   refs:      1,791,441,715
33 ==18642== I1  misses:            2,301
34 ==18642== LLi misses:            2,298
35 ==18642== I1  miss rate:          0.00%
36 ==18642== LLi miss rate:          0.00%
37 ==18642==
38 ==18642== D   refs:        853,671,979  (555,202,464 rd   + 298,469,515 wr)
39 ==18642== D1  misses:        4,036,351  (  1,055,017 rd   +   2,981,334 wr)
40 ==18642== LLd misses:        2,910,170  (      3,988 rd   +   2,906,182 wr)
41 ==18642== D1  miss rate:           0.4% (        0.1%     +         0.9%  )
42 ==18642== LLd miss rate:           0.3% (        0.0%     +         0.9%  )
43 ==18642==
44 ==18642== LL refs:           4,038,652  (  1,057,318 rd   +   2,981,334 wr)
45 ==18642== LL misses:         2,912,468  (      6,286 rd   +   2,906,182 wr)
46 ==18642== LL miss rate:           0.1% (        0.0%     +         0.9%  )
```

Listing 3: The annotated `cachegrind` output. (For the run-time output summary, see Listing 2.) Performance is given in terms of intruction or data cache reads and writes per function. Note lines 26 and 29 are two of the three most expensive operations in terms of cache misses for both read and writes, but—because they are 1-time initialization procedures—do not factor into the most critical hot loops.

```
1  --------------------------------------------------------------------------------
2  I1 cache:         67108864 B, 64 B, 2-way associative
3  D1 cache:         67108864 B, 64 B, 2-way associative
4  LL cache:         268435456 B, 64 B, 8-way associative
5  Command:          ./randwalk
6  Data file:        randwalk_test.cachegrind
7  Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
8  Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
9  Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
10 Thresholds:       0.1 100 100 100 100 100 100 100 100
11 Include dirs:     ~/code-hg/
12 User annotated:
13 Auto-annotation:  off
14
15 --------------------------------------------------------------------------------
16             Ir   I1mr  ILmr           Dr       D1mr  DLmr           Dw       D1mw       DLmw
17 --------------------------------------------------------------------------------
18 1,791,441,715 2,301 2,298 555,202,464 1,055,017 3,988 298,469,515 2,981,334 2,906,182  PROGRAM TOTALS
19
20 --------------------------------------------------------------------------------
21           Ir I1mr ILmr           Dr    D1mr  DLmr          Dw      D1mw      DLmw  file:function
22 --------------------------------------------------------------------------------
23 501,645,738    8    8 137,148,861     223    61 56,922,291        29         0  ???:__ieee754_expf
24 240,338,562    2    2  88,545,786       0     0 63,246,990         0         0  ???:fesetenv
25 188,439,137    0    0  47,137,234  79,641     0 19,408,216       141         0  moller_randwalk.f95:__moller_randwalk_MOD_randwalk_path
26 159,558,964    0    0  60,770,911 580,005     0 22,767,659   635,989   635,976  moller_randwalk.f95:__moller_randwalk_MOD_randwalk_preload
27 145,468,077    2    2  50,597,592       4     1 25,298,796         0         0  ???:expf
28 126,493,980    3    3  63,246,990       0     0 44,272,893        13         0  ???:feholdexcept
29 113,844,582    1    1  31,623,495       0     0 25,298,796         0         0  ???:fesetround
30  74,913,240    0    0  20,351,262 390,146     0 20,351,260 1,346,412 1,271,952  moller_randwalk.f95:__moller_randwalk_MOD_reflect_mass_asymm
31  65,682,948    0    0   7,892,244       0     0  1,315,374         0         0  mcnp_random.f95:__moller_randwalk_MOD_randwalk_path
32  39,539,501   11   11   9,410,084     121   117  9,045,079   996,819   996,818  ???:memcpy
33  31,623,500    0    0   6,324,700       0     0          0         0         0  ???:finitef
34  22,960,500    1    1   4,592,151       0     0          0         0         0  ../../../gcc-4.7.0/libgomp/config/linux/wait.h:gomp_barrier_wait_end
35  21,364,660    8    8   6,581,480       0     0  2,936,456         1         1  ../../../gcc-4.7.0/libgfortran/io/transfer.c:_gfortran_transfer_array
36  14,175,000    5    5   4,252,500       0     0  2,480,625         0         0  ../../../gcc-4.7.0/libgfortran/io/transfer.c:read_block_direct
37  14,047,182    6    6   3,937,256       0     0  2,165,311         0         0  ../../../gcc-4.7.0/libgfortran/io/unix.c:buf_read
38   8,505,000    3    3   2,480,625       0     0  1,771,875         0         0  ../../../gcc-4.7.0/libgfortran/io/transfer.c:unformatted_read
39   6,710,481  134  131   6,709,517      74     4        423         2         2  ???:???
40   4,592,173    2    2          17       2     0          0         0         0  ../../../gcc-4.7.0/libgomp/config/linux/x86/futex.h:gomp_barrier_wait_e
41   2,361,330    1    1     472,296       1     0          0         0         0  ../../../gcc-4.7.0/libgomp/config/linux/wait.h:gomp_team_barrier_wait_e
42   2,126,250    2    2   1,417,500       0     0          0         0         0  ../../../gcc-4.7.0/libgfortran/io/transfer.c:iolength_transfer
```

cache memory operations, and we can estimate which code operations are causing the greatest number of cache misses.

As another run-time tool, a couple of concerns outlined for `gprof` also apply to the profiles generated by `cachegrind`. First, the same code path concerns apply—we cannot draw any conclusions about code that was not executed during a particular run. Again, because our programs are single-purpose utilities, this caveat is not a critical concern for us. Similarly, the optimization passes performed by the compiler can also cause the correlation between machine instructions and source code to become unclear, so the same trade-off between production-equivalent executable and accurate results must be considered.

We gain the ability for Valgrind and `cachegrind` to provide information about the cache performance of our algorithms with a caveat—the simulator does not support OpenMP parallelization. The consequence of this deficiency is demonstrated in Figure 7; the performance likely fails to increase with the number of cores because memory and cache misses become a bottleneck in the algorithm. Despite the serial constraint, though, any poorly performing code identified in the serial case is only going to degrade as the memory bus is stressed during simultaneous access from multiple cores.

A.2.1 Program preparation & tool execution

No special preparation is required during compile time since the program is executed in a simulator which has access to all machine instructions, CPU, and memory internals at all times. Profiling is performed by executing the `valgrind` tool, instructing it to use the `cachegrind` tool, and specifying the program to profile. An example invocation may be,

```
$ valgrind --tool=cachegrind --cachegrind-out-file=randwalk.cachegrind \
> ./randwalk > randwalk.cachegrind-stdout
```

The file redirection produces the output shown in Listing 2, and use of the `cg_annotate` utility post-processes the file given by `--cachegrind-out-file` to produce a human-readable summary:

```
$ cg_annotate -I~/code randwalk.cachegrind > randwalk.cachegrind-annotate
```

The resultant annotated, human-readable output is shown in Listing 3.

A.2.2 Profile interpretation

The easiest results to interpret are shown in lines 41 and 42 of Listing 2. These two lines show that the simulated cache miss rate for reads and writes is less than 1%; from this we infer that the cache miss rate is not a significant contributor to the run time in the serial case. As has already been discussed in §6.2, though, this interpretation is likely not correct in the parallel case, particularly on SMP systems with a large number of cores.

More detailed information is contained within Listing 3. The sort order is by the number of instruction reads (analogous to the call count in `gprof`), and we see that the exponentiation function is called the most; note that the statistics are only for whole function calls unlike the output shown in Listing 1, and therefore we would expect the exponential function to be called often since it is used within the random walk inner loop. We are most interested in examining the columns marked with `mr` or `mw` in the names; these indicate the cache miss counts for the cache levels for both read and write operations, respectively. Lines 26, 30, and 32 immediately stand out. Both lines 26 and 30 do not pose a significant problem since these are both routines which must manipulate the entire energy matrix during the program initialization; a large number of cache misses are expected as caches are populated. The memory copy in line 32 is also expected to cause cache misses, but without further tooling and more specific profiling, we cannot conclude what causes the copies. Memory copies which occur due to loading data from disk is of no concern, but managing copies that occur during the random walk loop is an important task.

References

[1] Forrest Brown. *The MCNP5 Random Number Generator*. LA-UR-07-7961. Los Alamos National Laboratory. URL: https://laws.lanl.gov/vhosts/mcnp.lanl.gov/pdf_files/la-ur-07-7961_mcnp_rn.pdf (visited on 08/13/2013).

[2] Ashok Das and Thomas Ferbel. *Introduction to Nuclear and Particle Physics*. 2nd ed. World Scientific, 2009. ISBN: 978-981-238-744-8.

[3] Department of Computer and Information Science, University of Oregon. *TAU Performance System*. University of Oregon, Los Alamos National Laboratory, and Research Center Jülich, July 25, 2013. URL: http://www.cs.uoregon.edu/research/tau/home.php.

[4] Jay Fenlason. *The GNU Profiler*. The GNU Project, July 29, 2013. URL: http://www.gnu.org/software/binutils/.

[5] Brian Hayes. "Dividing the Continent". In: *American Scientist* 88.6 (2000), p. 481. DOI: 10.1511/2000.6.481. URL: http://www.americanscientist.org/issues/pub/2000/6/dividing-the-continent.

[6] Takatoshi Ichikawa et al. "Contrasting fission potential-energy structure of actinides and mercury isotopes". In: *Phys. Rev. C* 86 (2 Aug. 2012), p. 024610. DOI: 10.1103/PhysRevC.86.024610. URL: http://link.aps.org/doi/10.1103/PhysRevC.86.024610.

[7] Donald E Knuth. "Structured Programming with go to Statements". In: *ACM Computing Surveys* 6.4 (1974), pp. 261–301. URL: http://portal.acm.org/citation.cfm?doid=356635.356640.

[8] Peter Möller et al. "Heavy-element fission barriers". In: *Physical Review C* 79.6 (June 2009), p. 064304. DOI: 10.1103/PhysRevC.79.064304. URL: http://link.aps.org/doi/10.1103/PhysRevC.79.064304.

[9] Peter Möller et al. "Nuclear Ground-State Masses and Deformations". In: *Atomic Data and Nuclear Data Tables* 59.2 (Mar. 1995), pp. 185–381. DOI: 10.1006/adnd.1995.1002. URL: http://dx.doi.org/10.1006/adnd.1995.1002.

[10] Bogdan Povh et al. *Particles and Nuclei. An Introduction to the Physical Concepts*. Trans. from the German by Martin Lavelle. Springer, 2008. ISBN: 978-3-540-79367-0.

[11] Peter Ring and Peter Schuck. *The Nuclear Many-Body Problem*. Springer, 2004. ISBN: 3-540-21206-X.

[12] Vilen Mitrofanovich Strutinsky. ""Shells" in deformed nuclei". In: *Nuclear Physics* A122 (1968), pp. 1–33.

[13] *The OpenMP API specification for parallel programming*. The OpenMP Architecture Review Board, July 30, 2013. URL: http://www.openmp.org/.

[14] Valgrind Developers. *Valgrind*. July 25, 2013. URL: http://valgrind.org/.

[15] Luc Vincent and Pierre Soille. "Watersheds in digital spaces: an efficient algorithm based on immersion simulations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6 (1991), pp. 583–598. DOI: 10.1109/34.87344. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=87344.

[16] Samuels S.M. Wong. *Introductory Nuclear Physics*. second. John Wiley & Sons, Inc., 1998. ISBN: 0-471-23973-9.

Diffusion in Mixed Cells

(William Dai, mentor)

LA-UR-13-26470
Approved for public release; distribution is unlimited.

Title: Numerical Study for Diffusion in Material Mixtures Part I: Pure Materials

Author(s): Yeaton, Issac J.
Dai, William W.

Intended for: Computational Physics Summer Workshop, 2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)
Report

Issued: 2013-08-15

Numerical Study for Diffusion in Material Mixtures
Part I: Pure Materials

Isaac J. Yeaton*
Department of Mechanical Engineering, Virginia Tech

Mentor: William Dai†
Los Alamos National Laboratory

August 16, 2013

Abstract

The diffusion equation applied to material mixtures has been investigated for a variety of grid types and finite difference schemes. We present a conservative form for the finite difference equations that appropriately handles sharp material discontinuities for both cartesian and AMR grids. This formulation is then modified in *Part II* to deal with material mixing within a cell.

1 Introduction

The goal of this summer project is to investigate diffusion in material mixtures. More specifically, we are interested in what simplifications can be made to the thermal conductivity or governing equations to deal with this mixing and sharp discontinuities in material properties. The context for this project is Eulerian hydrodynamics/multiphyics codes.

There are two kinds of simulation for hydro. One type is based on a Lagrangian mesh, in which grid points move with the flow, and in the other type grid points are fixed. One of the major benefits of the Lagrangian approach is that originally the computational cells can be constructed so they are pure materials and then as the simulation progresses they remain pure. The major drawback with this Lagrangian approach is that the mesh can "tangle" where vorticity and high strain causes the mesh to deform to a point where the computation cannot converge. This is alleviated by remapping the deformed mesh onto something "nicer" so that the computation can continue. This remapping can lead to cells

*iyeaton@vt.edu
†dai@lanl.gov

that are no longer pure, but instead contain multiple materials. Remapping forms the basis of ALE (arbitrary Lagrangian/Eulerian) codes Galera et al. (2010).

One method to avoid this mesh tangling is to have a fixed mesh and have the fluid or material advect through it. This Eulerian approach is generally favored for fluid mechanics simulations because it avoids mesh tangling and easily handles vorticity. However, there is a major difficulty with using Eulerian methods for hydro problems with multiple materials: as the materials advect, mixed cells form where there is a subgrid interface between the two materials that is not resolved. How the interface is treated and what approximations are made can have implications for the physical relevance and accuracy of the simulation. This is especially true in the context of a multiphysics code, where each physics package may handle the interface separately, if it is handled at all.

There are a number of computational meshes that can be used for an Eulerian calculation, ranging from a regular, Cartesian grid to a completely unstructured one where each element has an irregular shape. A common mesh used, and one that we focus on in this study, is an adaptive mesh refinement (AMR) mesh. This is a regular mesh with a hierarchy of mesh sizes, or mesh refinement levels. Each grid element is square and refinement levels have an edge length half the length of the level above it. The mesh is not allowed to refine more than two levels at once; a cell must touch a refinement level one above or below its own at most.

The refinement portion of the mesh refers to increasing the resolution in areas where interesting physics occur, like near interfaces. Conversely, the mesh is coarsened in regions of homogeneous material or where no interesting physics occur. Figure 1 shows an example of an AMR mesh used in this study with four levels of mesh refinement near the circular material interface. Further from this interface are relatively coarse cells. We also made a regular mesh of completely fine cells to compare the solution to multiple materials when using the AMR grid. The benefit of the AMR mesh is a smaller number of cells; this AMR mesh has 2476 cells and the rectangular grid has 8836 cells, or an increase of 3.56 times.

The AMR mesh provides an effective way to deal with these material interfaces in certain circumstances. By repeatedly refining the mesh, the number of mixed materials in a cell decreases. However, it is impossible to refine all cells such that they are pure. The theoretical material interface can be reconstructed, resulting in an unstructured mesh Rider and Kothe (1998), similar to a Lagrangian mesh. This would provide a seemingly better estimate of the underlying physics. The location of this division is set so that the new area preserves the original volume fraction from the square cell. With the new cell boundaries calculated, the finite difference relations can be reformulated using pure cells on the unstructured mesh.

When doing this division, the mesh becomes more complex and more information is needed. The AMR mesh is very regular and it is easy to know neighbors, distances, and refinement information. This is not necessarily the case with an unstructured mesh and more information and complex data structures must be used. There is the the added complexity for solving diffusion because one now needs to know neighbor information,

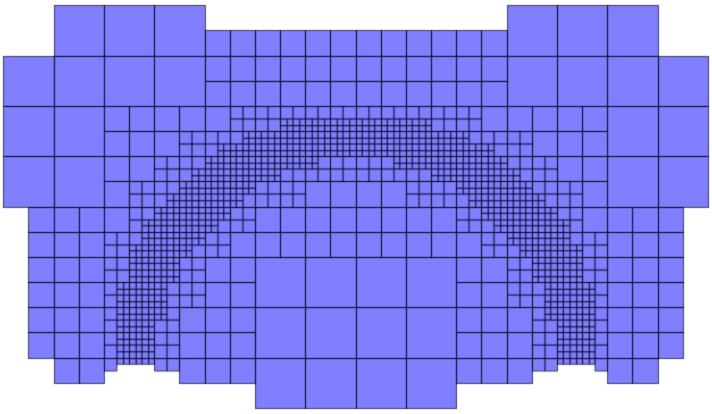


Figure 1: Sample of an AMR mesh where there is significant refinement at the material interface that is circular.

distances to centers of faces, and overlaps with neighboring cells. This is not needed with standard hydro and makes coding more difficult. A possible solution is to subdivide the cells, but instead of using the resulting unstructured mesh, the additional information is used to make a better estimate of the thermal conductivity on the original mesh. This is discussed in Part II of this report, detailing how to find the interface if there are only two materials, and how to formulate the finite difference relations.

A few comments here regarding this technique. While it uses this theoretical interface, caution must be taken if this new reconstruction accurately represents the physics. If there is spauling, where material is ejected, a straight-line interface may not accurately represent the real material. Additionally, if more than one material is present, or more material forms as the simulation progresses, how is this handled? The current technique uses gradients in volume fraction from neighboring cells, but if this new volume fraction only resides in one cell, the gradient is effectively zero and no interface is reconstructed. Another concern is how to order the multiple materials robustly, so that they accurately represent the physics (and so that heat transfer is not prematurely retarded if an insulator is placed in the wrong order). This becomes much more complicated in three-dimensions, where coding the interface reconstruction is a serious endeavor. Lastly, what if there is atomic mixing (not chunk mixing, where a piece of one material advected into the cell) and there is no experimental data on thermal conductivities in the mixture? These are just a few questions that have not yet been addressed in the current summer work, but some of the ideas presented here may be used as stepping stones to handle this mixing. Ideally, we can formulate a multi-material framework for dealing with mixing while maintaining the same mesh and making things easier to code and solve efficiently.

2 Flux based effective thermal conductivity

We now turn our attention to dealing with diffusion when there is no mixing, also known as pure or clean cells. The formulation for an effective thermal conductivity is based on temperature and flux being continuous at the material interface. To deal with the discontinuity, the conservative form of the governing equation is used and cell averages taken.

We are interested in equations of the following form (diffusion equations),

$$\rho c_p \frac{\partial T}{\partial t} = -\nabla \cdot \vec{F} + S, \tag{1}$$

where T is some quantity of interest, like temperature, F is the flux, and S is a source term. The flux is defined as

$$F \equiv -\kappa \frac{\partial T}{\partial x}, \tag{2}$$

where κ is the thermal conductivity, which is material dependent. For the following analysis, we assume κ does not change with temperature (so we are not considering radiation), but the following analysis holds if it were. Note that if $\kappa = \kappa(T)$, then there is an added complexity of now iteratively solving the matrix system and the nonlinear temperature equation. The flux is both space and time dependent, so a more general form of the equations is

$$\rho c_p \frac{\partial T(t, \mathbf{x})}{\partial t} = -\nabla \cdot \vec{F}(t, \mathbf{x}) + S(t, \mathbf{x}). \tag{3}$$

To derive the effective thermal conductivity, we consider one-dimensional heat diffusion using a uniform grid, as shown in figure 2, about the central cell. We assume the source term is zero, and integrate over the Δx domain and from $0 \leq t \leq \Delta t$, giving the integral form of the equations as

$$\int_{\Delta x} \int_{\Delta t} \frac{\partial T}{\partial t} dt dx = -\int_{\Delta t} \int_{\Delta x} \frac{\partial F}{\partial x} dx dt. \tag{4}$$

There is no approximation thus far and likewise we can perform the integrations without approximation as

$$\int_{x_{i-1}}^{x_i} u(\Delta t) dx - \int_{x_{i-1}}^{x_i} u(0) dx = -\int_0^{\Delta t} F(x_i) dt + \int_0^{\Delta t} F(x_{i-1}) dt. \tag{5}$$

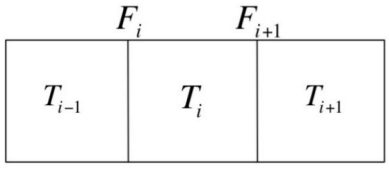


Figure 2: One-dimensional heat equation mesh.

Defining the following to simplify the notation,

$$u_i^n \equiv \frac{1}{\Delta x}\int_{x_{i-1}}^{x_i} u(\Delta t)dx$$
$$u_i^0 \equiv \frac{1}{\Delta x}\int_{x_{i-1}}^{x_i} u(0)dx$$
$$\bar{F}_i \equiv \frac{1}{\Delta t}\int_0^{\Delta t} F(x_i)dt$$
$$\bar{F}_{i-1} \equiv \frac{1}{\Delta t}\int_0^{\Delta t} F(x_{i-1})dt,$$

where we have spatially averaged the temperature in the cell, and temporally averaged the fluxes through the cell interfaces. We can now use this to write the discrete form of the equations, with some rearranging,

$$T_i^n = T_i^0 - \frac{1}{\rho_i c_{p,i}}\frac{\Delta t}{\Delta x}\left[\bar{F}_i - \bar{F}_{i+1}\right] + S_i\Delta t. \tag{6}$$

We use difference equations to calculate the flux based on the left and center cells,

$$F_i = \frac{-\bar{\kappa}}{\Delta x}\left(T_i - T_{i-1}\right), \tag{7}$$

The material constant $\bar{\kappa}$ is simply κ (the material thermal conductivity) if the materials are the same or the *effective interface thermal conductivity* if they are different. We can derive this $\bar{\kappa}$ based on conservation laws for the individual cells, namely that the interface cannot store or generate energy, so the flux leaving one cell must enter the other cell. We arrive at three equations that are all equal to describe this,

$$F_* = \frac{-\bar{\kappa}}{\Delta x}(T_i - T_{i-1})$$
$$F_{i,L} = \frac{-\kappa_L}{\Delta x/2}(T_* - T_{i-1})$$
$$F_{i,R} = \frac{-\kappa_R}{\Delta x/2}(T_i - T_*),$$

where F_* and T_* are the interface flux and temperature, respectively, and $F_{i,L}$ and $F_{i,R}$ are the fluxes from left and center cells, respectively. Rearranging, we get the effective thermal conductivity at the cell interface as

$$\bar{\kappa} = \frac{2\kappa_{i-1}\kappa_i}{\kappa_{i-1} + \kappa_i}. \tag{8}$$

This expression returns to the material thermal conductivity if both thermal conductivities are the same. Also, this was only derived for the flux entering from the left. The derivation holds for each interface and is easily extendable to two- and three-dimensions.

This method can also be used to find T_*, the interface temperature. Note that simply averaging the temperature between to adjacent cells gives an incorrect result. Similarly, in the derivation for $\bar{\kappa}$, averaging thermal conductivities between the neighboring cells, $\bar{\kappa} = 1/2(\kappa_L + \kappa_R)$, is also incorrect. This is a bad estimate and will not handle the interface correctly. It can be especially inaccurate if the two materials have very different thermal conductivities. However, since the diffusion equation tends to smooth things out, this error may go unnoticed, as the change is not necessarily obvious. Figure 3 shows the difference between the correct, flux based effective thermal conductivity, and the incorrect averaged method. Since the diffusion solve is global, the error at the interface will propagate through the rest of the solution domain.

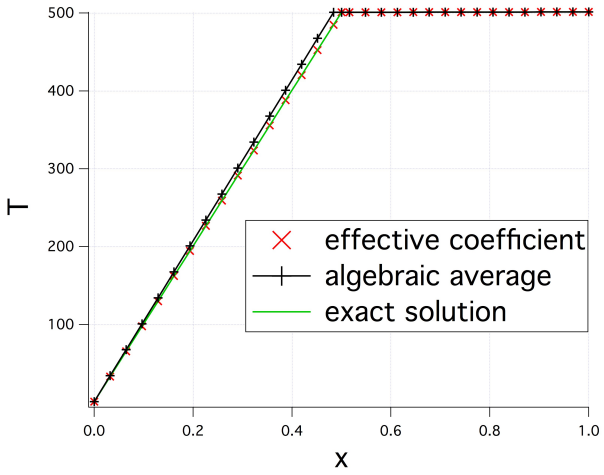


Figure 3: One-dimensional simulation for two material diffusion showing what happens when the interface is and is not handled correctly.

We can also formulate an effective interface thermal conductivity for an AMR mesh. This deals with not only the pure cell interface, but also when there are different mesh refinement levels. The general arguments are the same, but the flux has to be modified to take into account angles to the center of the small cell face. This is shown in figure 4. Using this, the thermal conductivity becomes

$$\bar{\kappa}_{\text{AMR}} = \frac{\kappa_i \kappa_k}{\frac{1}{4}\kappa_i + \frac{5}{8}\kappa_k}. \tag{9}$$

3 Time stepping method

A second-order accurate, implicit scheme was used to update to the new time step Dai and Woodward (1998a,b). This method has the benefit of going to the correct steady-state when

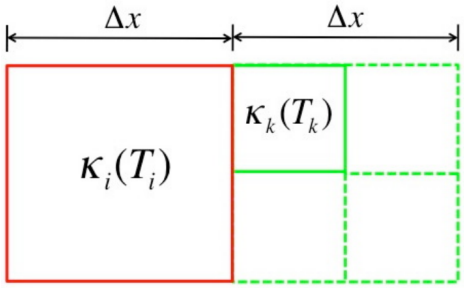


Figure 4: Flux for AMR mesh to find effective thermal conductivity.

the step size is large. Going to steady-state is beneficial when the thermal conductivities are very different. Depending on the time step, one material may diffuse heat completely (go to steady-state), while the other is still diffusing. Therefore it is important to go to steady-state correctly to alleviate these errors.

This method is formulated similar to a mid-point method and is graphically shown in figure 5. We use the flux at the half time-step to determine the temperature at the new time step. Likewise, the quarter time-step flux is used to calculate the half time-step temperature. Instead of doing a linear interpolation from the zero to half time step, since it would still depend on the old time step, a linear extrapolation from the full and half time steps is used. Therefore, everything is defined at new time steps and remains implicit. This is different from the Crank-Nicholson method, where the flux used to determine the new time step is define as

$$F \approx \frac{1}{2}\left(F^0 + F^n\right). \tag{10}$$

This will be unstable for large time steps, since the flux still depends on the temperature at the old time step (F^0).

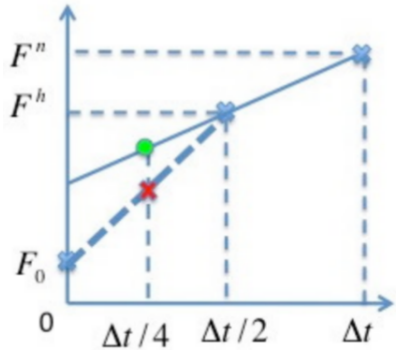


Figure 5: Graphical representation of time stepping scheme.

Using our flux definitions, the temperature difference equation (6) becomes

$$T_i^n = T_i^0 + \frac{\Delta t}{\Delta x}\left(F_i^h - F_{i+1}^h\right) + \Delta t S_i^h$$
$$T_i^h = T_i^0 + \frac{\Delta t/2}{\Delta x}\left(\bar{F}_i^h - \bar{F}_{i+1}^h\right) + \frac{\Delta t}{2} S_i^h,$$

where the superscript n signifies the new time-step and h is the half time-step. The flux approximation at the quarter time-step using the linear extrapolation becomes

$$\bar{F}^h \approx \frac{3}{2}F^h - \frac{1}{2}F^n. \tag{11}$$

Substituting this into the difference relations for the temperature we get

$$T_i^n = T_i^0 + \frac{\Delta t}{\Delta x^2}\left[\kappa_L T_{i-1}^h + \kappa_R T_{i+1}^h - (\kappa_L + \kappa_R)T_i^h\right] + S_i^h \Delta t$$
$$T_i^h = T_i^0 + \frac{3}{4}\frac{\Delta t}{\Delta x^2}\left[\kappa_L T_{i-1}^h + \kappa_R T_{i+1}^h - (\kappa_L + \kappa_R)T_i^h\right] + \frac{3}{4}S_i^n \Delta t$$
$$- \frac{1}{4}\frac{\Delta t}{\Delta x^2}\left[\kappa_L T_{i-1}^n + \kappa_R T_{i+1}^n - (\kappa_L + \kappa_R)T_i^n\right] - \frac{1}{4}S_i^h \Delta t.$$

We now have two coupled equations to solve for each cell in the mesh, which adds some additional computational work, but it has the aforementioned benefits.

For further clarity, we can develop the linear extrapolation function to get equation (11) as follows,

$$f = -2f^h\left(t - 1\right) + 2f^n\left(t - \frac{1}{2}\right), \tag{12}$$

where once again, the superscript n is for the full time-step and h is for the half. We can verify this is the correct expression by substituting in the known time steps and also the quarter time step,

$$t = 1 : f = 2f^n\left(\frac{1}{2}\right) = f^n$$
$$t = \frac{1}{2} : f = -2f^h\frac{-1}{2} = f^h$$
$$t = \frac{1}{4} : f = -2f^h\left(\frac{1}{4} - 1\right) + 2f^n\left(\frac{1}{4} - \frac{1}{2}\right)$$
$$= 2f^h\left(\frac{3}{4}\right) + 2f^n\left(\frac{-1}{4}\right)$$
$$= \frac{3}{2}f^h - \frac{1}{2}f^n.$$

4 Code Verification Study

Analytical solutions and the method of manufactured solutions (MMS) were used to test the code implementations. In MMS, we prescribe the desired temperature profile and then take the spatial and temporal derivatives as in equation (1). This differentiation results in an extra term, since it is unlikely a new analytical solution was found haphazardly. This term is then used as the source term in the governing equation. Therefore, we can now solve with an arbitrary initial temperature field and use the derived source term to get the exact solution we initially prescribed. This method was used to test both the convergence rate of the time and spatial components of the code. There was an issue getting the temporal part to converge on the exact solution, so an analytical solution in one-dimension was used. While this is a different implementation than used in the 2D code, it still tests the time stepping algorithm and how close it is to second order.

The exact temperature profile used for MMS is

$$T(x, y, t) = 1000 + 10 \sin\left(\frac{3}{2}\frac{\pi x}{L}\right) + 10 \cos\left(\frac{3}{2}\frac{\pi y}{L}\right), \tag{13}$$

where the temporal derivative is zero and the double spatial derivatives are

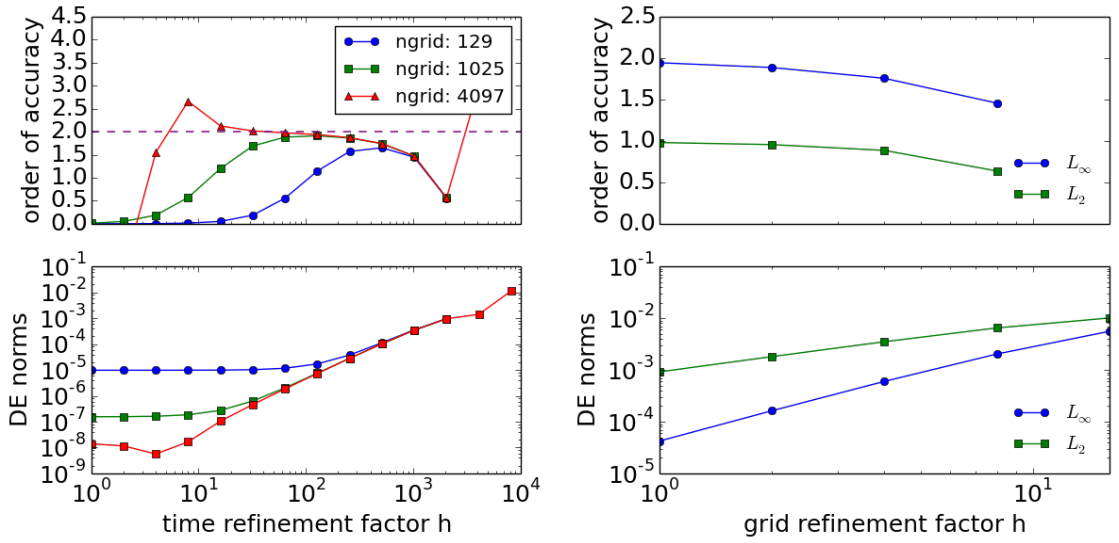
$$\frac{\partial T^2}{\partial x^2} + \frac{\partial T^2}{\partial y^2} = \frac{45}{2}\frac{\pi^2 \sin\left(\frac{3}{2}\frac{\pi x}{L}\right)}{L^2} + \frac{45}{2}\frac{\pi^2 \cos\left(\frac{3}{2}\frac{\pi y}{L}\right)}{L^2}. \tag{14}$$

Using equation (1), our source term S becomes

$$S(x, y, t) = -\kappa\left(\frac{\partial T^2}{\partial x^2} + \frac{\partial T^2}{\partial y^2}\right). \tag{15}$$

The open source symbolic mathematics library for Python, SymPy SymPy Development Team (2013), was used to derive the above expressions. Figure 6a shows the time convergence rate for the one-dimensional analytical solution. We expect second-order accuracy and we almost achieve that; it is formally $\mathcal{O}(\Delta t^{1.96})$. We believe this discrepancy is due to extrapolation step when finding the flux at the quarter time-step. We see artifacts of the mesh resolution affecting our convergence study. After the time-step has been refined about 100×, we see errors from the grid adversely affecting the solution, even through it has a very fine resolution. This causes the infinite error norm to level off and the observed order of accuracy to decrease to zero. The effect of the mesh on discretization error norms is also shown, where they flatten-out sooner. However, we do reach second-order accuracy.

For the grid convergence study, grids from 5×5 to 65×65 were used, with the grid spacing halved each time (figure 6b). For each grid, the CFL number ($\kappa \Delta t / \Delta x^2$) was maintained at 0.7 and the time step adjusted according for the given $\kappa = 0.05$. Each simulation was run until steady-state was reached and the max (infinite) and 2-norms were



(a) Time (b) Grid

Figure 6: Time and grid convergence plots.

taken in relation to the exact solution,

$$\text{err} = \left\| \frac{E - T}{E} \right\|, \tag{16}$$

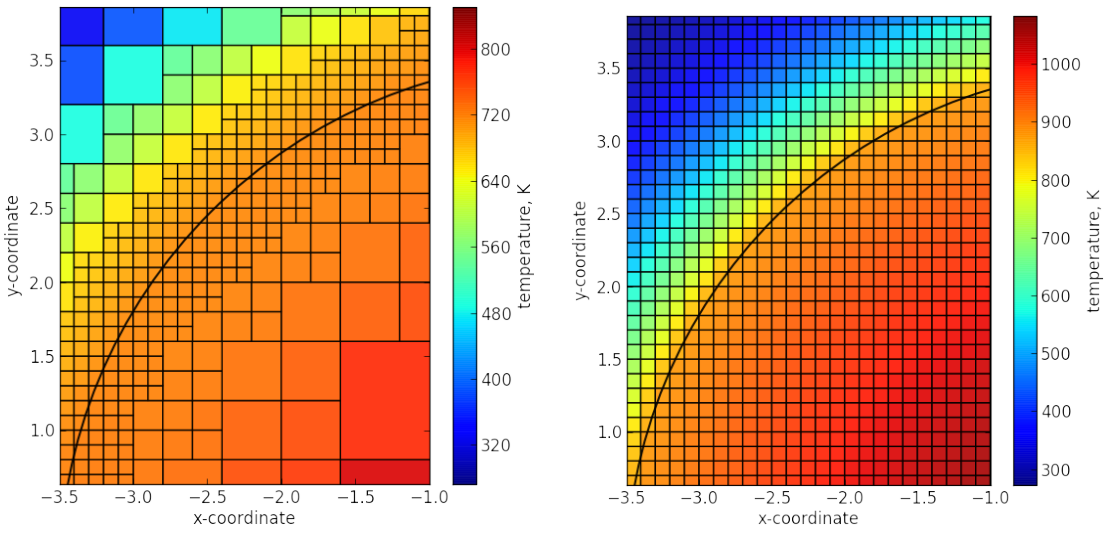
where E and T are the exact and calculated solutions, respectively. The discretization error norms were used to check the observed order of accuracy $\hat{p}$,

$$\hat{p} = \ln \frac{DE_{\text{coarse}}}{DE_{\text{fine}}} / \ln r, \tag{17}$$

where we use the discretization errors on the fine and coarse mesh and the mesh refinement r (which is two for this study). The grid refinement factor h is the grid spacing compared to the smallest grid spacing (a small h means the finest grid). As the grid is refined, we should see the observed order of accuracy approach two, which we do for the infinite norm. This is the norm generally used for diffusion since we often care about the maximum difference is, not the average. When the 2-norm is used, it shows first-order accuracy.

5 Sample results

We now present, in figure 7, a few results for pure cells for both AMR and cartesian meshes. This shows some of the capabilities of the code for handling different meshes and sharp material discontinuities. Both used thermal conductivities different by a factor of 7.24. The "theoretical" interface is shown in black and cells above/left have the higher thermal conductivity and cells below/right have the lower one. The cells are pure; cells with centroids on or outside the line are one material, cells inside are another. The cartesian mesh is composed of the greatest AMR refinement level and only a zoomed in portion (the upper left of figure 1) is shown. Effective interface thermal conductivities for both regular and AMR grids were used with the 2D tested implementation discussed above. The code uses GMRES, a Krylov subspace method, to solve the linear matrix equations, and sparse matrices to store the difference equations. A plot of the sparse matrix structure, when using the time stepping method discussed above, is shown in figure 8. From this diffusion study, we see that they look qualitatively similar, except that heat has diffused faster in the AMR case since a majority of the cells are larger. This is due to the larger cells on the exterior more rapidly changing temperature.



(a) AMR (b) Cartesian

Figure 7: AMR and cartesian mesh runs with pure material interfaces and thermal conductivities between materials different by a factor of 7.24.

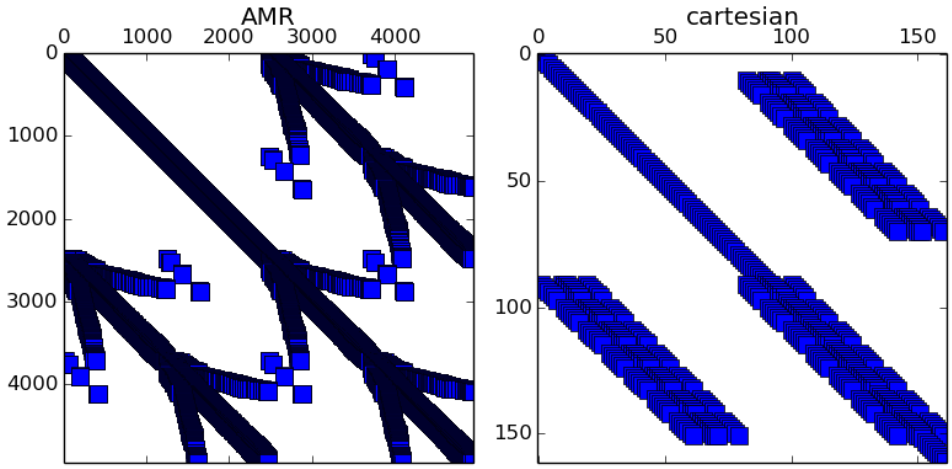


Figure 8: Sparse matrices for both AMR and cartesian meshes for types problem solved in figure 7

6 Conclusion and Future Work

A physics based formulation for the finite difference diffusion equation has been presented that can accommodate an arbitrary material discontinuity. A second-order accurate scheme in both space and time, with the implicit time scheme going correctly to steady-state, has also been discussed and implemented. Additionally, preliminary results for dealing with material interfaces using AMR has been shown. While the majority of the framework has been laid to begin experimenting with in-cell mixing, it can still not handle AMR grids. An example of a volume fraction field on an AMR grid, like would be found in a multi-physics simulation, is shown in figure 9. Next steps would be to apply interface reconstruction to this mesh and compare with the pure cells above. Additionally, a fully unstructured solver should be coded as a way to deal with the irregular geometry introduced by splitting the cells. This could provide the "true" solution from which other methods of dealing with mixing can be tested against. This would complete the testing framework to deal with material mixing and interfaces.

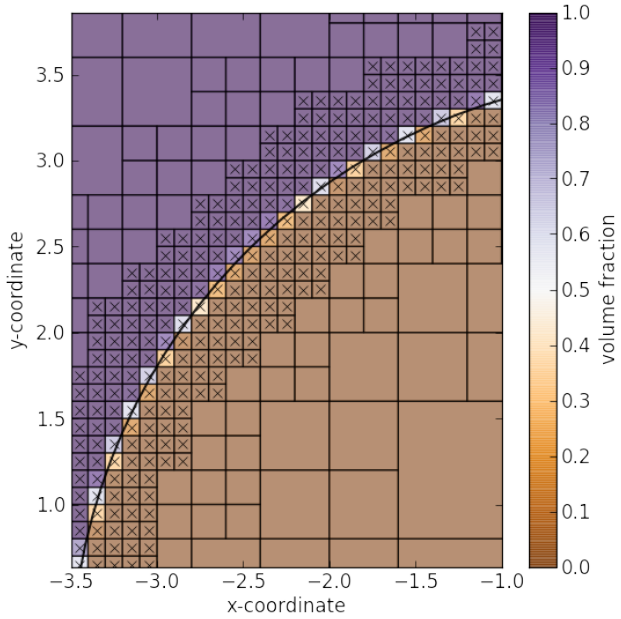


Figure 9: Volume fraction for AMR mesh with known, continuous material interface.

One of the main difficulties of dealing with these mixed cells is reconstructing the interface. The current implementation only handles two materials in 2D and relies on the standard gradient method (Young's method) Rider and Kothe (1998). This becomes difficult with more materials, where their ordering becomes important. One way to deal with this and leverage previous work is to look to the computer vision community. Here the goal

is generally fast identification of interesting features in an image, like finding boundaries or "segments" of an image. This allows for compression of certain regions and increases computational speed Achanta et al. (2012). When making these segments or superpixels, it is possible to have subpixel interfaces using some algorithms Forte; Malmberg et al. (2009). If we view the computational mesh as an image, where volume fraction for a particular material can be a color intensity (for example, RGB could correspond to three different materials), it maybe be possible to use one of these algorithms. The subpixel interface would correspond to the sub-cell material interface that we cannot resolve. Lastly, the ordering of particular materials can be dealt with by stacking different meshes (as they progress and change in time) into a three-dimensional structure where voxel interfaces are calculated Chang and Tao (2010).

This idea is very nascent, but it might prove a new application of computer vision and leverage their advanced algorithms for dealing with interface identification. However, there are a few items that need to be addressed first. One is a mapping from mesh coordinates to a regular grid to use the algorithms. Another is making sure that the constructed interface conserves mass/area after the reconstruction. An additional constraint to the algorithms may be needed to ensure this. Lastly, this does not solve one of the most difficult problems of finding the overlapping areas between adjacent cells. There might be ways around this by making approximations using the new cell centroid, which should be easy to find from the interface reconstruction. This could be an extension of ideas presented in *Part II*.

References

Radhakrishna Achanta, Appu Shaji, Kevin Smith, and Aurelien Lucchi. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 34(11):2274–2281, 2012.

Ming-Ching Chang and Xiaodong Tao. Subvoxel Segmentation and Representation of Brain Cortex Using Fuzzy Clustering and Gradient Vector Diffusion. pages 76231L–76231L–11, March 2010. doi: 10.1117/12.843853. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=747654.

Wenlong Dai and Paul R Woodward. Numerical simulations for nonlinear heat transfer in a system of multimaterials. *Journal of Computational Physics*, 139(1):58–78, 1998a.

Wenlong Dai and Paul R Woodward. Numerical simulations for radiation hydrodynamics. i. diffusion limit. *Journal of Computational Physics*, 142(1):182–207, 1998b.

P. Forte. A SIMPLE METHOD OF SEGMENTATION WITH SUBPIXEL ACCURACY. pages 403–405.

Stéphane Galera, Pierre-Henri Maire, and Jérôme Breil. A two-dimensional unstructured cell-centered multi-material ALE scheme using VOF interface reconstruction. *Journal of Computational Physics*, 229(16):5755–5787, August 2010. ISSN 00219991. doi: 10.1016/j.jcp.2010.04.019. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999110001956.

Filip Malmberg, Joakim Lindblad, and Ingela Nystr. Sub-pixel Segmentation with the Image Foresting Transform. *Combinatorial Image Analysis*, pages 201–211, 2009.

William J. Rider and Douglas B. Kothe. Reconstructing Volume Tracking. *Journal of Computational Physics*, 141(2):112–152, April 1998. ISSN 00219991. doi: 10.1006/jcph.1998.5906. URL http://linkinghub.elsevier.com/retrieve/pii/S002199919895906X.

SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2013. URL http://www.sympy.org.

LA-UR-13-26644
Approved for public release; distribution is unlimited.

Title: Numerical Study for Diffusion in Material Mixtures: The Treatment of Mixed Material Cells

Author(s): Roberts, Thomas M.

Intended for: Computational Physics Summer Workshop, 2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)
Report

Issued: 2013-08-22

Numerical Study for Diffusion in Material Mixtures: The Treatment of Mixed Material Cells

T. Maximillian Roberts
Mentor: William Dai
Los Alamos National Laboratory
Computational Physics Workshop, Summer 2013
(Dated: August 14, 2013)

In the modeling of hydrodynamics on an Eulerian mesh, the interface between different material types can be advected in a manner such that a cell on this mesh may contain multiple material types. Material properties such as conductivity for pure materials are well known and tabulated for use in computation, but these mixtures will have material properties which may not be well represented by these empirical values or even some averaging of them. This project focused on the treatment of these mixed cells by approximating the sub-cell structure from gradients in the material volume fraction. Using information from this "interface reconstruction", the sub-cell structure was used to improve approximations of directional conductivity. Finally, this method was compared to a simple averaging of material properties in the mixed cells.

I. INTRODUCTION

The heat equation, a specific case of the diffusion equation, is one of the most elementary partial differential equations. Given in differential form:

$$c_p \rho \frac{\partial T}{\partial t} = -\nabla \cdot \vec{F} = \nabla \cdot (k \nabla T) \tag{1}$$

where c_p is the thermal heat capacity, ρ is the mass density and k is the thermal conductivity.

For our interests, we will be focusing on the role of thermal conductivity in this equation, so for the rest of this discussion we will take the product of heat capacity and density to be unity for all materials in question. As such the equation now only involves one coefficient which we will continue to refer to as k, the conductivity (which is generally called the thermal diffusivity).

For the simple case of k being a constant across the domain of interest, with reasonable boundary and initial conditions, an analytic solution is easily obtained through separation of variables. In 2D, for the case of Dirichlet boundaries, $\delta\Omega = 0$, and initially constant temperature of u_0, we have:

$$u(\vec{x}, t) = \sum_{m,n} A_{mn} \sin(\frac{m\pi}{L}x) \sin(\frac{n\pi}{L}y) e^{(-k\frac{(m^2+n^2)\pi^2}{L^2}t)} \tag{2}$$

$$A_{mn} = \iint u_0 \sin(\frac{m\pi}{L}x) \sin(\frac{n\pi}{L}y) dx dy \tag{3}$$

Note higher order terms decay exponentially faster, so after a short time only the lowest order solutions play a significant role.

We can consider the discrete form of the heat equation that we will numerically solve. If k is constant on the domain we can pull it out of the derivative. Looking first at the RHS, we can consider the Laplacian, which we know can be discretized in various forms. Using the second order centered differencing scheme, we approximate:

$$\nabla \cdot (k \nabla T) = k \nabla^2 T \approx k \frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2} \tag{4}$$

Consideration of the time derivative is more involved. An explicit scheme would take the terms on the RHS to be from the previous time, t^n, while an implicit scheme takes these terms to be from the next step, t^{n+1}. Explicit solvers are simple, but are fundamentally constrained by the CFL limit for stability. We will use an implicit time step:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = k \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} \tag{5}$$

and we solve this expression for T^{n+1}.

Solving this equation numerically shows the expected behavior which can be compared to the analytic solution. Note that this method has first order accuracy in time but second order in space. This is shown easily by plotting error from the solution calculated with increasing resolution in time or space on a log-log plot. An example solution is shown in Figure 1.

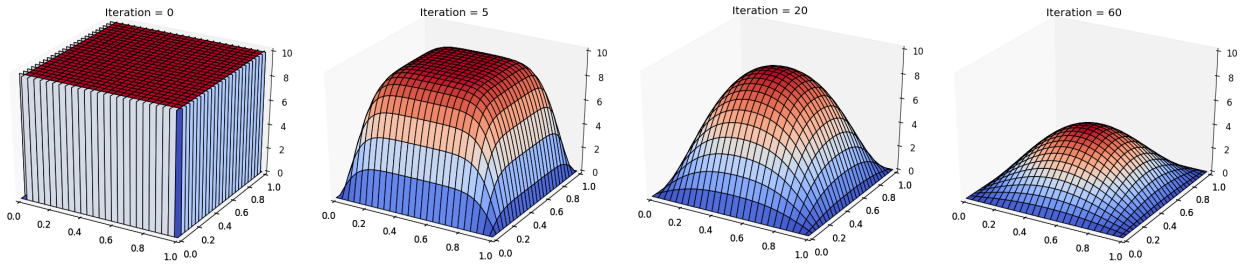


FIG. 1: Solutions to the heat equation with boundaries fixed at zeros and initially constant temperature on the domain.

II. MATERIAL MIXTURES

So far we have discussed the case of domains where the materials type is constant. If material properties are not constant in space, then k cannot simply be pulled out of the derivative. This makes solving the heat equation much more challenging. If we consider the case of two materials joined at a sharp interface, then we can have a discontinuity in the material properties in the domain. In the differential equation, this discontinuity would result in a singularity which is clearly not physical.

While there are certain tricks one can play via variable transformation and other methods, an analytic solution to this problem is difficult and very specific. From physical intuition we can ballpark what the solution will look like. Lets take the 1D case of two materials initially at a uniform temperature, cooled from both ends. If the left material has a higher conductivity than the right, we would expect heat to leave faster through the left side than the right. As such, the left material would cool quickly, resulting in more rapid cooling of the right material as well. This is illustrated in Figure 2. We know that where the two materials contact they should be at the same temperature, and that the flux into the left material from the interface must equal the flux out of the right material on that side. Therefore both temperature and thermal flux are continuous across the interface. This basic idea is a good sanity check, but for quantitative analysis we will need to do much better.

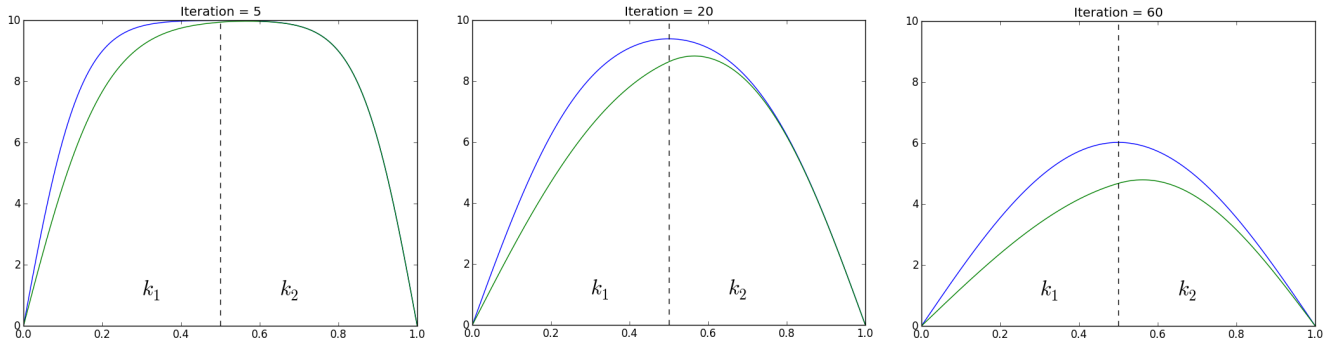


FIG. 2: Comparing the expected solutions between the pure material solution (blue) and the mixed material where the left half has twice the conductivity as the right.

We would like to note at this point that for this project we focused on thermal diffusion on an Eulerian mesh, a mesh that is fixed in the "lab" frame of reference. This is in contrast to a Lagrangian mesh, which is a mesh that moves with the material. As we are dealing Eulerian meshes, it is likely that if there is advection in the simulation, cells with different material types will mix. This will result in cells containing multiple values for conductivity. For bookkeeping of materials on a sub-cell scale, the fraction of each material in the cell is recorded, known as the "material volume fraction." This volume fraction is known for each material type in each cell. We will from here on refer to "pure" cells as having a volume fraction of 1 for a specific material, and "mixed" cells as having multiple material types. Note that all information pertaining to the structure of that material in the cell is lost in using a volume fraction. The material could be evenly diffused throughout the cell or lie on a sharp interface.

At this point we can now begin to ask some questions; How we treat these cells with mixed materials? What does one use for the conductivity of this cell? Is a simple averaging of the properties of two materials sufficient? Could we do something better? These are one of the main questions this report will begin to address. The next section begins to tackle the problem, starting with pure cell interfaces.

III. EFFECTIVE CONDUCTIVITY, $\tilde{k}$

As we saw above, the handling of a discontinuity in the differential form of the heat equation is non-trivial. How do we discretize this appropriately? By ignoring the fact that a change in the material conductivity affects the heat flow out of the discrete form (effectively pulling k outside the derivative), we will be discarding important physics. We could perform a simple averaging of adjacent cell properties, but on what grounds can we justify this?

We can handle the discontinuity in material properties more physically by considering the flux between two adjacent cells of different materials. We know two things at this interface; the temperature infinitely close to the interface on both sides must approach the same value, and the heat flux into the interface must equal the flux out of the interface, which must equal the flux across the interface (assuming no surface or point source). As such, we have just created a statement about the heat flux, and therefore conductivity, between these two cells based on a physical law. The situation is depicted in Figure 3.

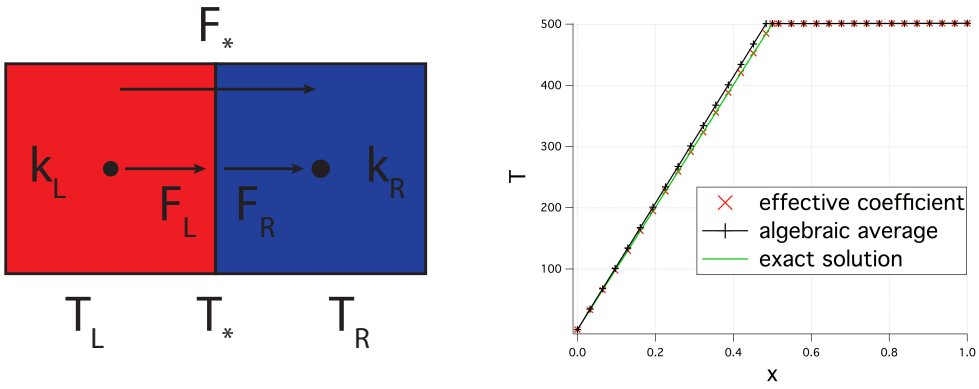


FIG. 3: Left: Two cells containing different materials. Conservation of flux at the material interface yields an appropriate way to express the conductivity between the different materials. Right: A comparison between the algebraic mean and the effective conductivity method for a 1D test case between two materials. Figure used courtesy of William Dai.

From this simple statement of conservation of flux we can write the following four statements:

$$F_L = -\frac{k_L}{\Delta x/2}(T_* - T_i) \qquad F_R = -\frac{k_R}{\Delta x/2}(T_{i+1} - T_*) \tag{6}$$

$$F_* = -\frac{k_*}{\Delta x}(T_{i+1} - T_i) \tag{7}$$

$$F_L = F_* = F_R \tag{8}$$

From these statements, we can solve for the temperature T^* at the interface, as well as the effective conductivity, $\tilde{k}$, between the two cells. These are given as:

$$T_* = \frac{k_R T_{i+1} + k_L T_i}{k_L + k_R} \qquad \tilde{k} = \frac{2k_L k_R}{k_L + k_R} \tag{9}$$

Note the expression for $\tilde{k}$ is a sort of "reduced conductivity", or in limit of $k_1 << k_2$, the conductivity behaves most like the smaller conductivity. Also notice that in the case of equal material properties the two cells conduct with the natural conductivity of their material. T^*, while we don't use it here, also has its applications.

Comparing a solution computed using this effective conductivity to one that simply averages conductivity between neighboring cells, we see the effective conductivity is much more representative of the "true" solution, as is shown on

the right side of Figure 3[1].

As such, we have implemented this effective conductivity in our code to best represent the conductivity between cells of different materials. Note that this method is limited to describing pure cells of one material type or another. The case of mixed material cells is more complicated and will be discussed next.

IV. INTERFACE RECONSTRUCTION

A. Basic Principles and Method

Given a material volume fraction on a mesh, one can use gradients in this volume fraction to approximate the interfaces between different materials. For instance, lets take the situation where we have two materials with a transition between them. Each material is represented by a fractional value of the total material at that point, the material volume fraction. If we take the gradients of one material's volume fraction, we know that the surfaces of constant volume fraction must lie perpendicular to these gradients.

On this mesh, we will have pure cells of each material type, and at the transition there will be mixed cells. If this transition occurs quickly enough (spatially), we can approximate a material interface in these mixed material cells as a line perpendicular to the volume fraction gradient. This line will divide the mixed cells and will be placed based on the cell's volume fraction. Performing this for all the mixed cells will approximate and interface between the materials at a sub-cell scale.

For this project we weren't given a test case, so we created our own volume fractions. To represent a material on a mesh, we placed ones and zeros in a matrix to form various test material interfaces. The presence of a one meant that cell was pure of the material type in question, a zero meant there was none of that material in the cell. To create mixed cells we simply took block averages on this mesh. For example, if we started with an $N \times N$ mesh of ones and zeros, by averaging cells in blocks of $n \times n$, we returned a smaller matrix of dimensions $(N/n) \times (N/n)$ where there are now ones, zeros and fractional values at the material interface. This was our artificial mixing process. For illustration, an example of this is shown in Figure 4.

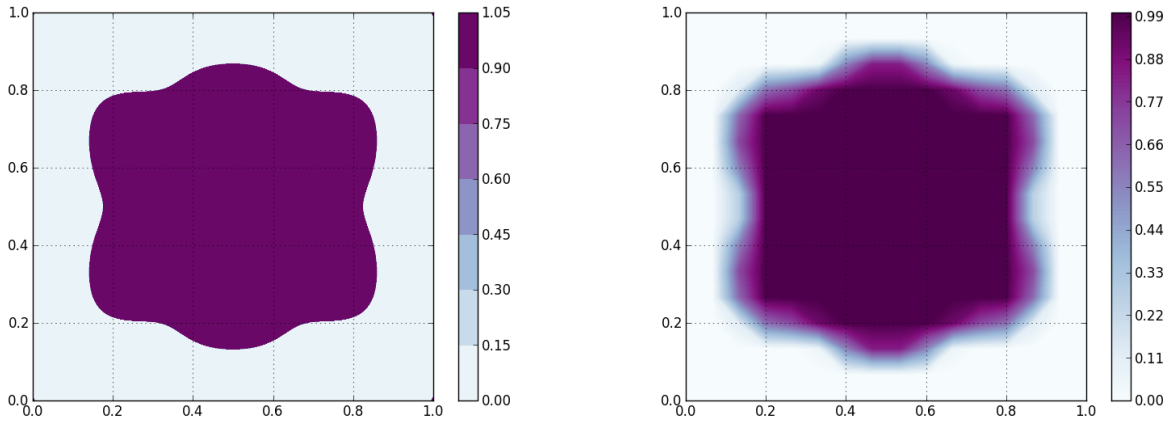


FIG. 4: Initial clean material interface on the right (ones and zeros on matrix) and the artificially mixed volume fraction on the left.

B. Volume Fraction Gradient and Interface Line

To find gradients of volume fraction on the mesh, we used a linear least-squares method. For a given cell, by fitting a line in a least-squares sense to the values for volume fraction in the surrounding cells, in both the x and y directions, the slopes of these lines yield the components of the gradient in volume fraction. By performing this for the entire mesh we produce the leftmost plot in Figure 5.

---

[1] Figure courtesy of William Dai.

In these mixed cells the interface will be a line perpendicular to the cell's volume fraction gradient. Clearly, the line perpendicular to the gradient vector goes as the negative reciprocal of the gradient vector components $(-x/y)$. These initial interface lines are shown in the rightmost plot in Figure 5.

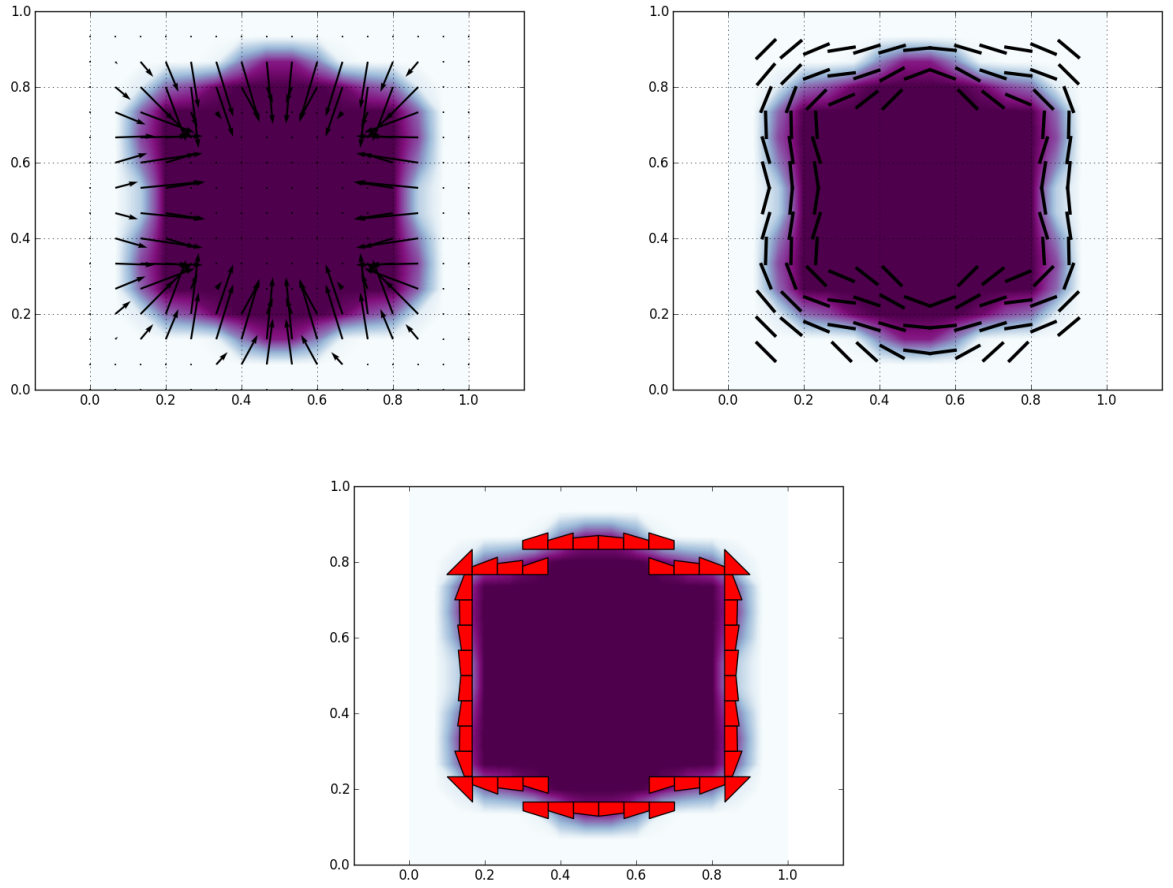


FIG. 5: The process to make the initial interface: find the gradients in the volume fraction, from these gradients find the perpendicular lines in each cell. In the cells with fractional volume fraction, use these perpendicular lines to create interface cells.

To begin to form our interface, what we call "interface cells" are created by dividing the cell by using the line perpendicular to the cell's gradient vector. Initially, we divide the cell in half simply by defining the line to pass through the cell center. We perform this for cells that have a non-zero and non-unity volume fraction only. These initial cells are shown in the bottom of Figure 5.

These "interface cells" we create maintain the important information about the cells that are not pure. They are objects that store the cells material properties, corners, area, edge information and other details necessary for later calculation. The positions of the dividing line ends and the corners of the cells are stored as the "nodes" of the cell. Depending on the angle of the interface line, the number of nodes can be 3, 4 or 5.

C. Interface Cell Area Resizing

The last step in recreating the interface requires moving each interface line based on the cell's volume fraction. The angle of the line is fixed as it must remain perpendicular to the gradient, but its position in the cell (y-intercept) can change such that the fractional area "under" the interface line equals the cell's volume fraction. To change the area of the cell, we simply need to move the positions of the two nodes which describe the line dividing the cell (the interface line). The manner in which we move these nodes depends on whether we are increasing or decreasing the cell's current area and by how much.

Next, in each case we need to know how much to increase or reduce. For only small changes in cell area, the interface line may only need to be moved slightly thereby not significantly changing the cell structure. By moving the interface line too far, one of the nodes of this line will hit the corners of the cell. After this, this node will now need

to move along the other edge of the cell. After this happened, the cell will have either gained or lost a corner and an edge. For these cases the cell object must be redefined such that we properly manage the shape of the cell for later calculation. An example of this process is shown in Figure 6.

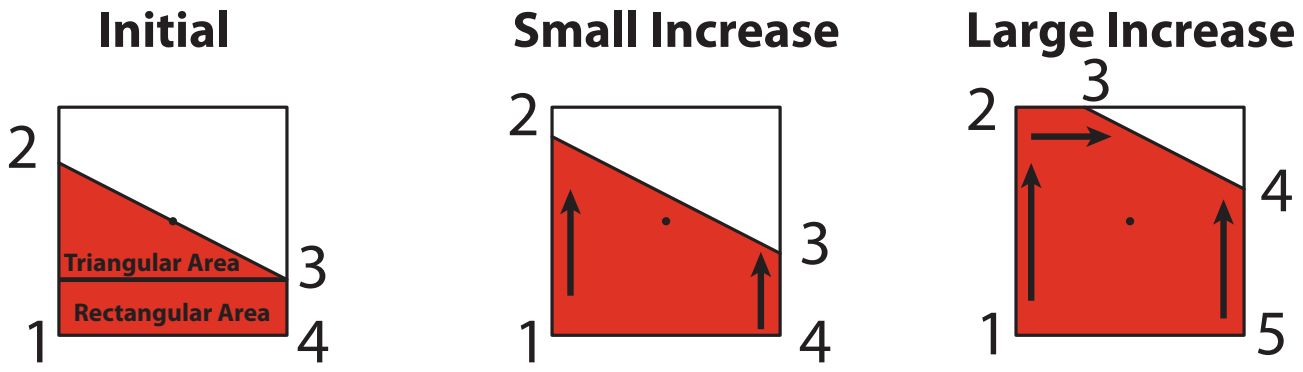


FIG. 6: Depending on the cell's volume fraction the nodes describing the interface line must be moved in different ways. For a small enough change the nodes of the cell may simply be repositioned, but past a critical value the nodes must be redefined, adding a node and edge. An interface cell can be divided into its "triangular" and "rectangular" regions which allows for a simple means of determining how to resize the cell based on it volume fraction and gradient angle.

We can discuss the cell area in terms of its rectangular and triangular regions, as is shown in the first cell in Figure 6. Using this description, we find an algorithm for area resizing as follows:

- Volume fraction less than current fractional area (need to reduce area):
  - Volume fraction less than triangular area:
    - Completely remove rectangular cell area, reduce triangular region appropriately.
    - Note we now need to remove a node and edge in this case.
  - Volume fraction greater than triangular area:
    - Need to reposition the existing interface nodes accordingly, simply reduce the rectangular area.
- Volume fraction greater than current fractional area (need to increase area):
  - Volume fraction greater than triangular area plus twice the rectangular area:
    - Increase cell area such that only a triangular area is not occupied, reduce this triangle.
    - Note we now need to add a node and edge in this case
  - Volume fraction less than triangular area plus twice the rectangular area:
    - Need to reposition the existing interface nodes accordingly, simply increase rectangular area.

Note that for each above case, the dividing nodes are assigned based on gradient angle, so we need to handle the various ranges of angle for each case. This resizing code required a substantial amount of consideration and debugging.

D. Finding Sub-Cell Centers and Edges

After rescaling these interface cells we need information about various geometric features. We will see later the position of the interface cell centroids will be important, so calculating these is necessary. To simplify this calculation, a python package called Polygon was used. This package has several function specific to the treatment of polygons, and from the nodes of each interface cell Polygon objects were created. Use of this package allowed for simple plotting of these cells as well as knowledge of the cell area (to confirm the resizing) and the cell center, which is simply stored as an (x, y) coordinate in the cell.

The other important detail the interface reconstruction gives us are the edges of these interface cells. The edges of each cell are stored as simply two points (or nodes) of the cell's edge. A convention was enforced to maintain a left, right, top, bottom order to the arrangement of nodes and edges. The left side is the starting side, with the lowest left point being the starting node. Left is given priority over bottom as is shown in Figure 7.

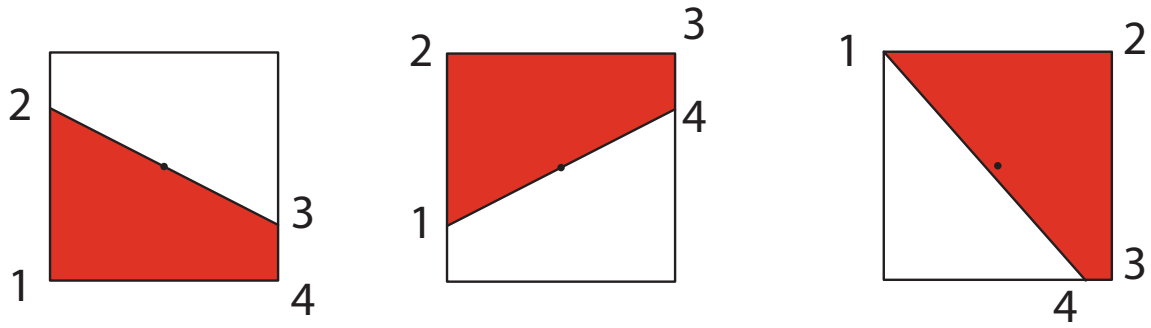


FIG. 7: Left is given highest priority as the initial node, lower left higher than upper left. The node order proceeds clockwise.

E. Resulting Interface

After performing an interface reconstruction using the above method for both volume fractions, an approximation to the material interface becomes clearly defined. In each mixed material cell an interface line divides the two material types. Generally in the case of a sharp interface (only a one or two mixed cells between pure cells), this reconstructed interface is relatively representative of the original interface, as in shown in Figure 8.

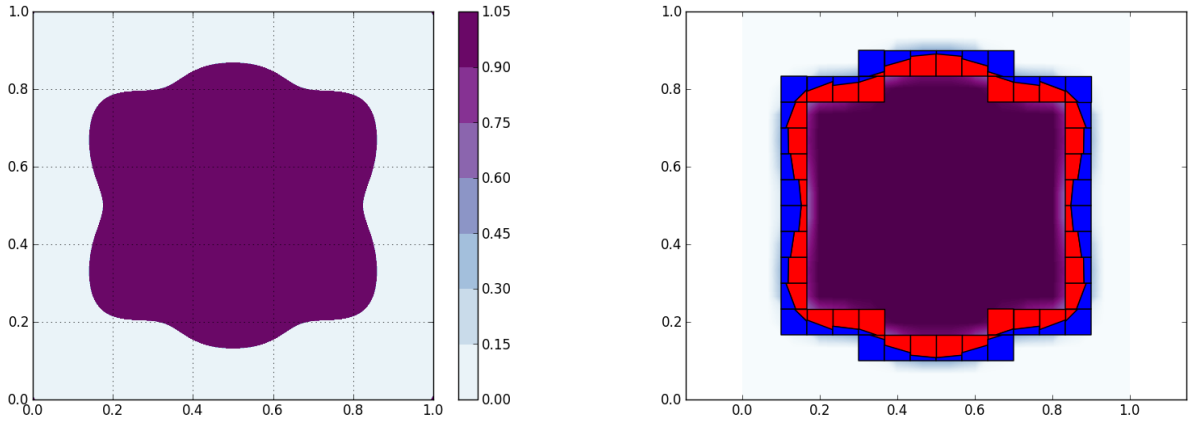


FIG. 8: By performing the interface reconstruction on both material volume fractions we have an approximation for the positioning of the materials on a sub-cell scale. While the new interface represents the original, note that it is discontinuous between cells.

The interface is discontinuous as a line was used for the interface in each cell. This line was constrained first to be perpendicular to the gradient in volume fraction (slope), then placed by the cell's volume fraction (y-intercept). As such, there is no more freedom in the fit to make the interface continuous. The use of a higher order polynomial or some other method could possibly be employed to smooth this interface, but that was beyond the scope of this work.

It is also important to mention that this method as described above fails when more than two material types are present at an interface. This method currently works as the gradients of different material volume fractions are anti-parallel. If three or more materials were present, this condition would most likely not be satisfied and there would be resulting overlap of interface cells in the reconstruction. This is clearly unphysical, and therefore a fundamental limitation of the aforementioned technique. Other codes handle this problem by putting certain constraints on the arrangement and alignment of materials on the mesh[1]. For example, there may be a requirement that the materials are always arranged in a certain order, or that the materials are always aligned based on the direction of some primary material species.

V. USING THE INTERFACE RECONSTRUCTION

From the interface reconstruction described in the last section, we can now approximate the location of materials in these mixed cells based on gradients in the volume fraction. Given this interface, we could easily divide the mixed cells each in two, yielding a mesh of only pure materials. In one way this greatly simplifies the issue; given only pure cells we already have a method to calculate the conductivity based on conservation of flux, which is shown to be an accurate representation. While these cells are no longer interacting across a purely vertical or horizontal interface the same underlying principle holds, so the calculation would simply be adjusted geometrically. The real issue is that if we divide these cells we are forced to move from the mesh we started with to a mesh that can handle the unstructured cells resulting from this interface. This poses a severe computational burden as a regular grid can be handled with much faster methods.

We wanted to somehow use the information from the interface reconstruction to improve the accuracy of the calculation on the original mesh, a way of representing the division of the cells on the interface while using only a single cell for the two materials. By using the information of the placement of the different material types in the cell, we hoped to find an alternative representation of the conductivity into these cells that would best represent the two divided cells. We do this by considering the directional effective conductivities.

A. Deriving Effective Conductivity with Cell Center of Mass

Given a cell divided as shown in Figure 9, we can discuss the flux from the upper sub-cell through the right boundary into the neighboring pure cell:

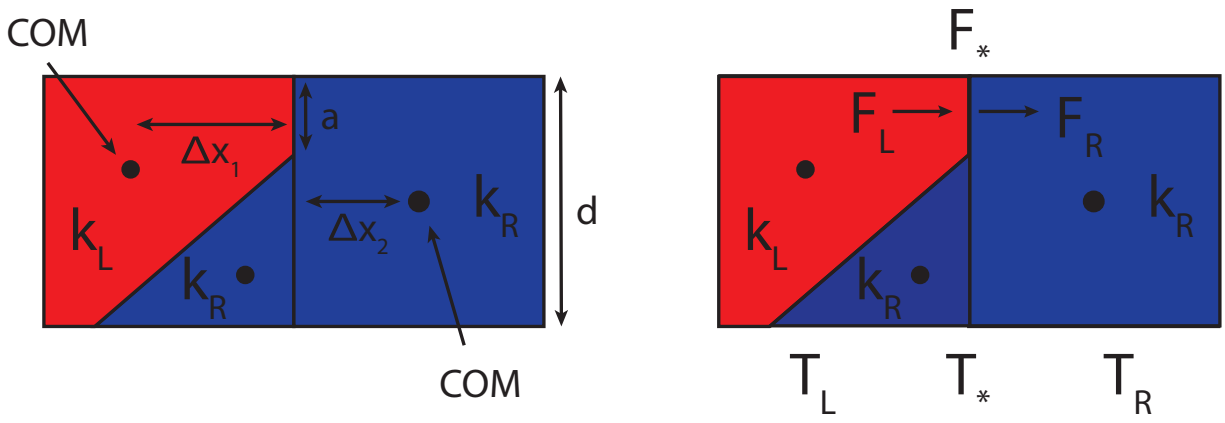


FIG. 9: Two neighboring cells, one divided by the interface reconstruction. On the left we show the sub-cell centroid geometry and the fractional flux boundary between the cells. On the right the representation to derive the effective conductivity is illustrated.

From the same arguments of conservation of flux as used to find the previously derived effective conductivity, we can find write the following expressions:

$$F_L = -\frac{k_L}{\Delta x_1}(T_* - T_i)\frac{a}{d} \qquad F_R = -\frac{k_R}{\Delta x_2}(T_{i+1} - T_*)\frac{a}{d} \tag{10}$$

$$F_* = -\frac{k_*}{\Delta x_1 + \Delta x_2}(T_{i+1} - T_i)\frac{a}{d} \tag{11}$$

$$F_L = F_* = F_R \tag{12}$$

Notice in these we use the sub-cell centroid for determining the flux through the boundary, but as the flux is dotted with the normal to the surface as given by:

$$\frac{\partial T}{\partial t} = -\nabla \cdot \vec{F} \rightarrow \int \frac{\partial T}{\partial t} dV = -\int \nabla \cdot \vec{F} dV = -\oint \vec{F} \cdot \hat{n} dA \tag{13}$$

we see that only the x separation is important in this case. Here we will use the approximation that these sub-cell temperatures are equivalent to the original cell-center temperature, though later we discuss that this approximation should be improved. Under this assumption, we find the effective conductivity between these neighboring cells to be given as:

$$\tilde{k} = \frac{k_L k_R}{\frac{k_L}{\Delta x_1} + \frac{k_R}{\Delta x_2}} \frac{\Delta x_1 + \Delta x_2}{\Delta x_1 \Delta x_2} \tag{14}$$

Note that this expression returns to the original expression for effective conductivity between two pure cells in the case that $\Delta x_1 = \Delta x_2$.

To find the actual effective conductivity between two of these interface cells, we must consider the flux between all neighboring sub-cells. Above we found the new effective conductivity, but the flux for a specific k is only through a portion of the boundary as is shown in Figure 9. As such, we compute the effective conductivity through each sub-cell boundary and scale this based on the fractional portion of the cell edge the sub-cell is in contact with. These scaled conductivities are then summed:

$$F_{total} = \frac{a}{d} F_{above} + \frac{d-a}{d} F_{below} \tag{15}$$

$$= -\frac{a}{d} \tilde{k}_{above} \nabla T - \frac{d-a}{d} \tilde{k}_{below} \nabla T \tag{16}$$

$$= -(\frac{a}{d} \tilde{k}_{above} + \frac{d-a}{d} \tilde{k}_{below}) \nabla T \tag{17}$$

$$= -\tilde{k}_{tot} \nabla T \tag{18}$$

B. Cell-Center Approximation

As mentioned earlier, we are attempting to represent the information from the interface reconstruction about two cells effectively in one. By considering the conductivities through sub-cell edges we have begun this process, but are still missing part of the representation. This approximation currently uses the same value for both sub-cell temperatures, the cell-center temperature. In many instances this will be a poor approximation. We can't afford to keep track of two temperatures as we wish to maintain the original mesh style, but we can find a better approximation for the sub-cell temperature. Methods considering the sub-cell centroid as well as the internal conductivity of the cell based on our interface reconstruction have been discussed but not thoroughly developed as time in the workshop was limited.

VI. COMPARISONS TO CELL AVERAGING

Using the interface reconstruction to calculate the effective conductivity for the interface cells and their neighbors we have now have a new means of describing the heat transfer between these cells. Earlier we had discussed cell averaging based on volume fraction to give us a simple approximation to a cell's conductivity. As such, we have two methods to compare. We use a test case of two materials separated by at a circular interface. The interior material is of lower conductivity than the external material. First, we can look at the actual differences in effective conductivity directly. If we break a cell into four quadrants as shown in Figure 10, we can represent the effective conductivities with neighboring cells more easily.

Using this visualization we can compare the representations of the interface conductivities. In Figure 11 we show the effective conductivity using the cell averaging method and using the interface reconstruction. Superimposed on these plots is a black circle representing the actual material interface.

Comparing these two plots, immediately one notices that the interface reconstruction sharpens our representation of the material conductivity. This comes from a better representation of the conductivity based on the materials in contact between cells. For example, notice the cells at the top of the "interface method" plot. The top of the mixed cells are made of outer material, and the conductivity between this outside boundary and the above pure cells is based only on the outer material's conductivity. Compare this to the same cells in the "cell averaging" plot. We see here that the averaging of the materials in the mixed cells changes the effective conductivity between two regions that should conduct uniformly. Cell averaging appears to diffuse the actual material interface.

To more quantitatively compare these methods we need a "true" solution to compare to. In contrast to the single material case, we don't have an analytic solution. Alternatively we could compare results found by dividing the mesh along the interface, but this requires coding an unstructured mesh method which we unfortunately didn't have time to

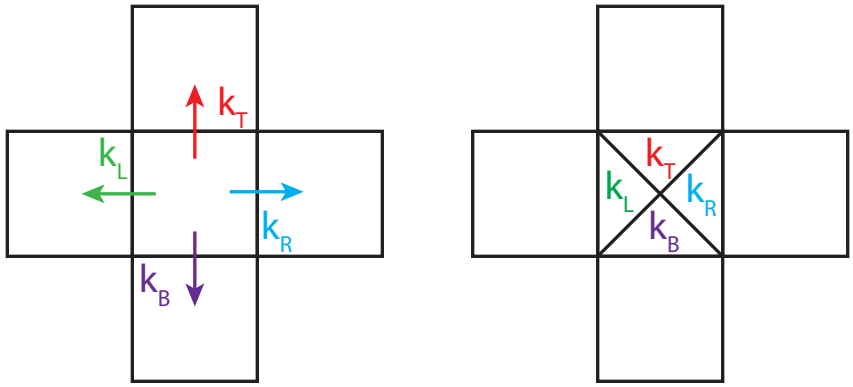


FIG. 10: For visualization we divide the cell into four quadrants to represent the effective conductivities with the cell's neighbors.

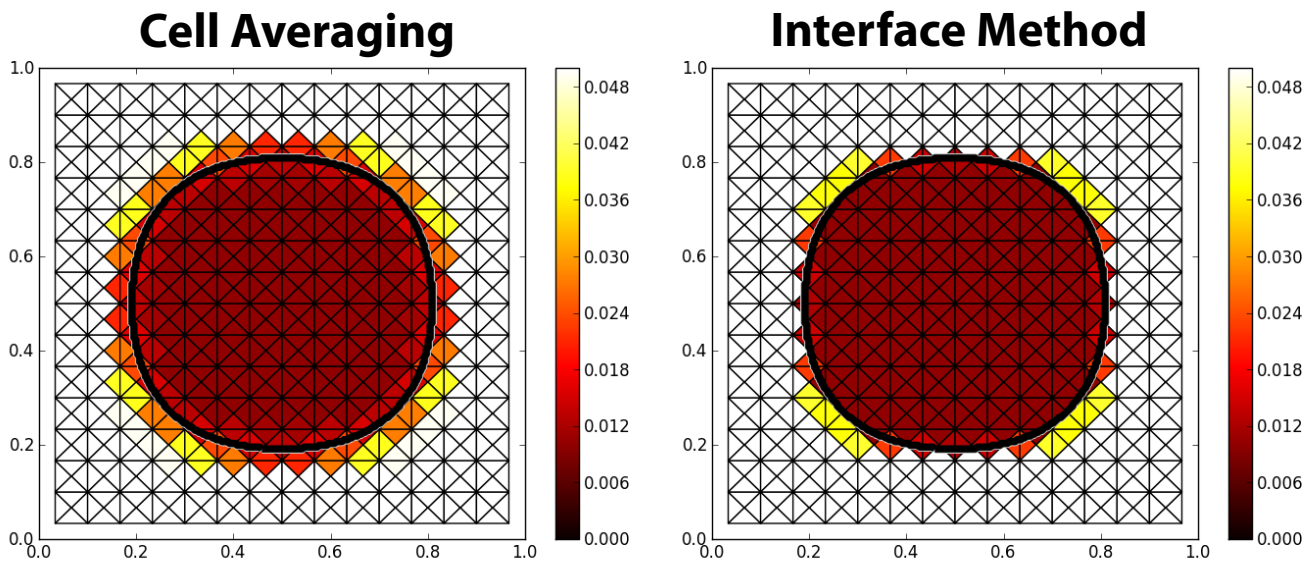


FIG. 11: On the left is a visualization of the effective conductivity from using a cell averaging of conductivity based on volume fraction. On the right is the result if you use information from the interface reconstruction instead. The black ring represents the actual material interface. Notice the sharpening of our representation of the conductivity using the interface method compared to the cell averaging.

do. As a last resort we can perform a high resolution run where the original interface we artificially mixed to produce our volume fraction is used as the true interface. This is performed simply using the pure cell effective conductivity first discussed.

Using the true solution described above, the error for both methods was calculated. Various ratios between the two material conductivities were tested. This is plotted for both methods in Figure 12.

As Figure 12 shows, there is a clear trend with ratio between the material conductivities for the interface method, but no clear trend for the cell averaged method. For small ratios between k_1 and k_2 we see that the cell averaging method seems to be the more accurate of the two, but for conductivities over an order of magnitude different the interface reconstruction method becomes more accurate. This answers the earlier question; if the materials are significantly different conductors, when they are averaged it may be a poor representation. Note that the two methods agree completely for the unity ratio case.

If we look at contour plots of the error from both methods, we gain some insight into what is being misrepresented. Figure 13 plots the difference from our true solution for increasing k ratio.

You'll notice that the interface reconstruction method always underestimates the temperature of the interior material while the cell-averaging method initially underestimates, then overestimates for large enough conductivity. The structure of the cell-averaging error changes dramatically from $k_1/k_2 = 2$ to $k_1/k_2 = 10$, then is very consistent for larger proportionality. This suggests that the conductivity averaging process changes the effective "structure" of the conductivity in a less predictable way than the interface reconstruction method.

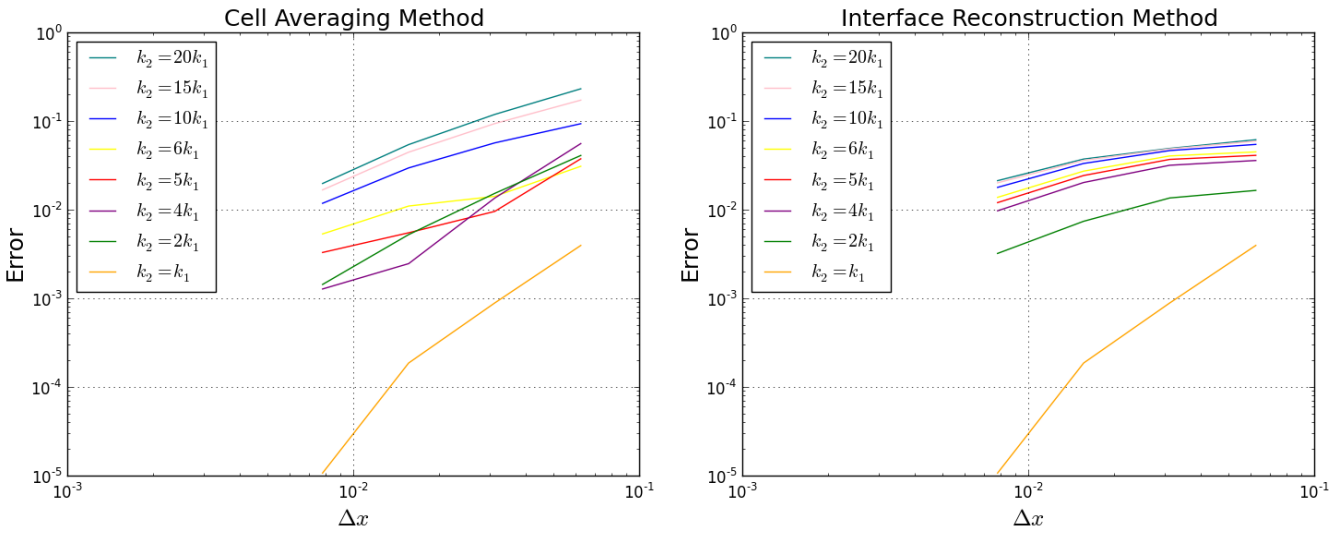


FIG. 12: L2 norm error for both the cell-averaging method and the interface reconstruction method with grid size. 8 ratios between the conductivities are plotted, showing that the cell averaging technique is more accurate for more similar materials, but the interface reconstruction is better for materials with significantly different conductivities.

The fact that interface method effectively cools the interior faster than it should while the cell averaging method cools the interior too slowly can be directly related to how the methods represent the material interface. As the cell averaging diffuses the interface, lower interior conductivities are spread out, effectively insulating the interior material. Opposite to this, the interface method sets the effective conductivity of an interface cell equal to the outside conductivity, even if the cell is mostly inside the interface. This cools the interior material faster than it should, an issue which could be accounted for by more properly considering the sub-cell temperatures.

VII. CONCLUSION

We have seen the method using the interface reconstruction to improve our estimations of conductivity between cells is in fact better than simple averaging of the material properties in a mixed cell for the case of significantly different conductivities at a material interface. For cases where the conductivities of the materials were similar, cell averaging did prove to be more accurate though the strange effect of the ratios between the conductivities showed that this method of may have issues. It was also clear that the interface method always underestimated the interior material temperature for the test geometry we used.

To improve on this current method, some means of describing the mixed cells internal mechanisms should be imposed. Currently both sub-cells use the cell-center temperature as the temperature at their centroid which is a poor approximation. Going through the derivation of effective conductivity considering a more proper treatment of the sub-cell center temperature yields a more complex result requiring some interpolation to find this sub-cell temperature based on neighboring cell-center values. The algorithm involved to account for this has been conceived, but the timeframe of the project prevented the implementation. It seems reasonable to believe that by considering these sub-cell temperatures properly, the results from the interface method should be as good if not better than the cell averaging method for all ratios between conductivity.

With regard to application of this method, one obvious example is at the finest resolutions of AMR style meshes separating two materials. To completely represent the material interface an unstructured mesh would be necessary, but maintaining the AMR type mesh is certainly computationally advantageous. These unstructured meshes will become particularly taxing when we move from 2D to 3D. As an alternative the interface reconstruction could mostly represent this interface while maintaining the original mesh structure. While the interface reconstruction coded for this project was 2D, a 3D version can certainly be envisioned though it may be substantially more tedious to code. It should be noted that while in 3D the interface reconstruction could prove to be relatively expensive, reconstruction is clearly a local operation and is therefore easily parallelizable.

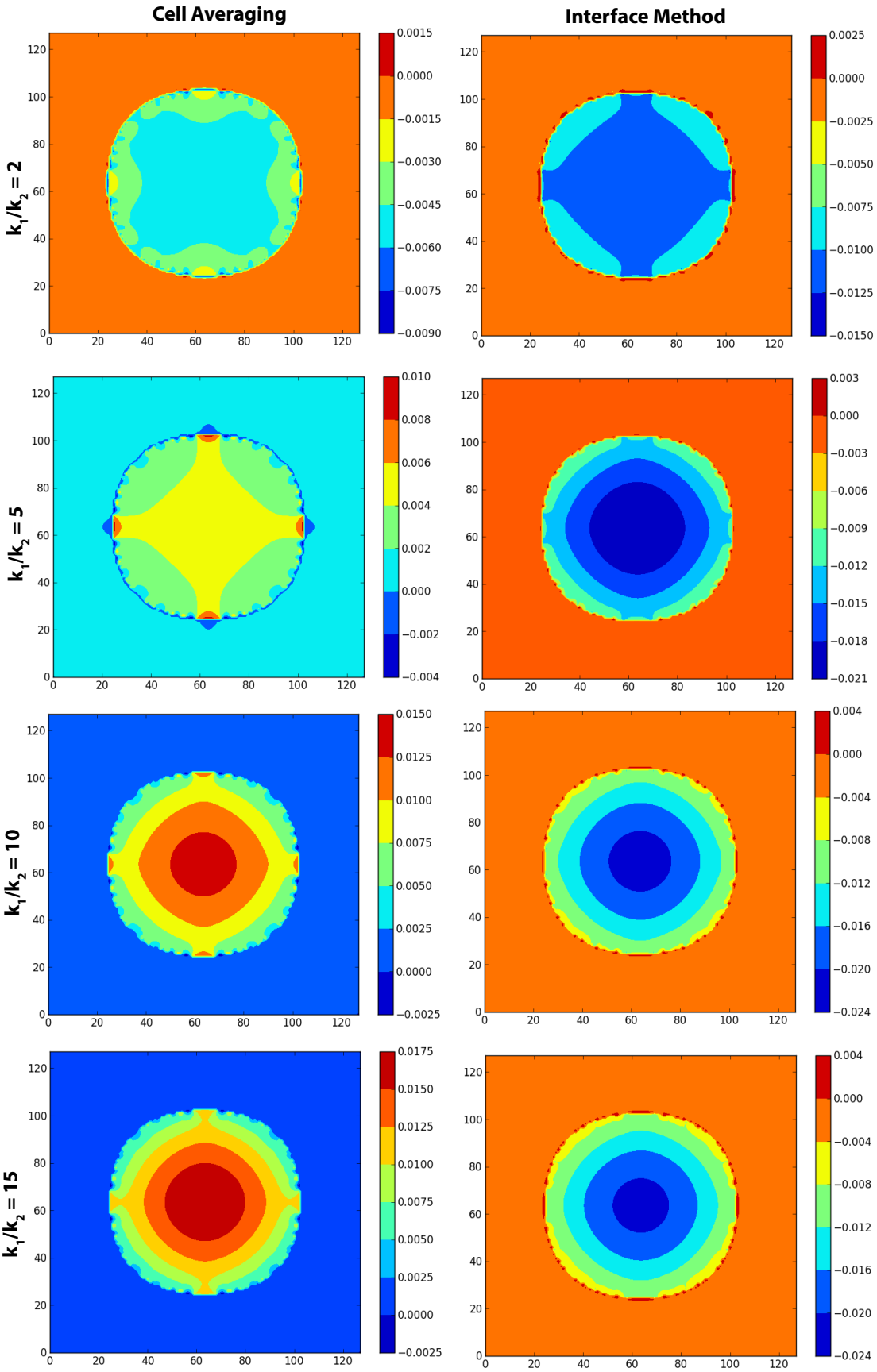


FIG. 13: Plots of difference between "true" solution and solutions from both methods for increasing k ratio. Notice that the overall behavior for the interface method is more consistent, even for smaller k ratios. The interface method always underestimates interior temperature while the cell averaging method often overestimates.

VIII. AKNOWLEDGEMENTS

I would like to thank my mentor William Dai for his advice and weekly lecture-style meetings where we learned the error of our ways. We were given more freedom than most projects with regard to direction and how to handle problems, which made the project highly educational. I would also like to thank Scott Runnels for organizing this workshop and taking the extra time to make this experience memorable and fun!

---

[1] W. J. Rider, D. B. Kothe, *Reconstructing Volume Tracking*, JOURNAL OF COMPUTATIONAL PHYSICS 141, 112-152 (1998), ARTICLE NO. CP985906

Turbulence Modeling

(Dan Israel and John Schwarzkopf, mentors)

BHR Equations with Immiscible Effects: Preliminary Work

Jeremy A. Horwitz, John D. Schwarzkopf

Abstract

Compressible turbulent flows with dispersed phase effects are found in many applications ranging from combustion to cloud formation. However, these types of flows are among the most challenging to simulate. While the exact equations governing a system of particles and fluid are known, computational resources limit the scale and detail that can be simulated in this type of problem. Therefore, a common method is to simulate averaged versions of the flow equations, which still capture salient physics but which are relatively less computationally expensive. Besnard et al. [1] developed such a model for variable density miscible turbulence, where ensemble-averaging was applied to the flow equations to yield a set of filtered equations. Besnard et al. further derived transport equations for the Reynolds stresses, the turbulent mass flux, and the density-specific volume covariance, to help close the filtered momentum and continuity equations. We re-derive the exact BHR closure equations including integral terms owing to immiscible effects. Physical interpretations of the additional terms are proposed. The goal of this work is to extend the BHR model to allow for the simulation of turbulent flows where an immiscible dispersed phase is non-trivially coupled with the carrier phase.

Table of Contents

Introduction

Variable density turbulent flows with dispersed phases are ubiquitous in engineering applications ranging from combustion in automobile and aircraft engines to environmental applications such as cloud formation. In flows involving multiple materials, we identify the material which has the largest contribution to the system's dynamics as the carrier phase and materials having higher order contributions to the flow dynamics as dispersed phases. Such identification is typically suitable when the carrier phase volume fraction is much larger than all dispersed phase volume fractions. However, in many flows even at low dispersed phase volume fraction, dispersed phase dynamics may have a contribution comparable to that of the carrier phase dynamics. In such cases, the presence of a dispersed phase may non-trivially couple with carrier phase dynamics to produce more complicated flow physics. When the dispersed phase material is immiscible with respect to the carrier phase (i.e. the phases do not mix at the molecular level), additional physics including the treatment of boundary conditions (slip, penetration, sources of mass, interfacial tension e.g.) must be accounted for since these effects may have a significant contribution to the total system's dynamics. Specifically, the addition of rigid particles to a turbulent fluid flow may either increase dissipation of turbulent kinetic energy (TKE) or enhance production of TKE depending on the particle size relative to the turbulent integral scale [13]. As the volume fraction increases, particles can no longer be treated as independent. In this regime, turbulent dissipation owing to particle-fluid interaction is accompanied by particle dissipation owing to particle-particle interaction in the form of collisions [10]. In regions where multiple particles are separated by scales comparable to or smaller than the un-laden Kolmogorov scale, the laden-dissipation length scales may be altered [16]. In flows involving droplets or bubbles, there may exist mass transfer between the dispersed and carrier phase in the form of evaporation or condensation. Mass transfer in turn affects production of TKE and turbulent mass flux (as we will show). When density variations and/or compressibility in the carrier phase is significant, additional production in the Reynolds stress equation may occur. Therefore, the combination of compressibility and dispersed phase coupling with the carrier phase makes simulating compressible turbulent flows with immiscible dispersed phases especially challenging. Particularly, the range of length scales in practical flows of this type makes direct numerical simulation (DNS) unviable.

A breakthrough in modeling variable density multicomponent turbulent flows came in 1992 when Besnard et al. [1] applied ensemble averaging to the instantaneous mass, momentum, and energy equations and derived exact transport equations for the Reynolds stresses, the turbulent mass flux, the density-specific volume covariance, and the turbulent heat flux. These exact equations were valid for compressible turbulence involving multiple miscible fluid components. The turbulent mass flux and density specific covariance transport equations were derived as a means for closing the Reynolds stress equation. With suitable closure models for the remaining terms in the Reynolds stress equation (dissipation, triple-correlation, pressure-strain, transport terms), as well as for the unclosed terms in the remaining transport equations, this system of

equations are collectively referred to as the BHR model which is capable of capturing compressible single phase multicomponent effects for many flows [2]. Introduction of the BHR equations is postponed to the discussion section.

Independently, a large body of work exists on incompressible particle-laden flows where the carrier phase is treated as a single uniform material. In 1850, Stokes [4] first derived the drag on a spherical particle moving at a constant speed in a steady fluid in what would later be identified as the low-Reynolds number or creeping flow limit. From the Oseen equations, Kaplun and Lagerstrom as well as Proudman and Pearson derived a second-order correction to the non-dimensional drag coefficient corresponding to the Stokes solution [6]. In 1888, Basset [5] used successive approximations to derive the equation of motion for a spherical particle beginning from rest and falling slowly due to gravity in a viscous fluid. Batchelor [6] discusses the role of oscillating boundary layers on mean fluid element motion and shows that a mean drift of fluid elements or small particles will exist if there are spatial gradients in either amplitude or phase of the flow away from a domain boundary. Unsteady effects and curvature in the flow velocity are accounted for in the particle equation of motion derived by Maxey and Riley [7]. Later, Maxey [8] examined the motion of particles in a turbulent flow and showed that inertial particles should experience a net flux into regions of low vorticity and high strain-rate. Squires and Eaton [9] performed a DNS of particle-laden turbulence and observed the preferential concentration effect predicted by Maxey [8]. Sundaram and Collins [10] observed enhanced turbulent dissipation due to particle presence where the peak transient dissipation increased with particle Stokes number. Burton and Eaton [11] performed a fully-resolved DNS of a turbulent flow over a single particle and compared the force on the particle with a similar expression to that derived in [7]. The dominant contribution to the particle-drag is shown to be the steady Stokes drag and all other drag effects are small for almost all times. The discrepancy between the theoretical drag and calculated drag suggests the particle-force is more complicated in instances where the particle diameter is of order Kolmogorov scale or greater [11].

The challenge of studying particle-laden turbulence may be eclipsed by the difficulties in understanding flows with droplets and bubbles. In such flows, the dispersed phase is no longer rigid and may deform subject to the stresses exerted by both the carrier phase as well as the dispersed phase when the volume fraction of the latter is not small. At material interfaces between disparate immiscible molecular species, interfacial tension will play a dominate role in flow dynamics in the neighborhood of the interface when the appropriate non-dimensional parameter is not large (Weber, Capillary, Bond number e.g.). Attempts to model surface tension have had some success, (see [12] for early work on the subject). However, simulating surface tension and interfacial dynamics remain a challenging subject. We will not address this topic further except to mention in our results where such a model would be present. In the case of deformable droplets surrounded by a gaseous carrier phase of the same material, (e.g. water droplets in water-vapor) mass-transfer between the two phases may exist under the appropriate

thermodynamic conditions. Hereto, we will not present an adequate review except to point out in our discussion section which terms are owing to mass transfer effects and to briefly discuss simple models which may capture the salient aspects.

Having briefly examined some of the relevant literature and additional physics present in variable density turbulent flows with immiscible dispersed phases, we state the objective of this work. In this paper, we present an extension to the equations first derived by Besnard et al. [1]. After a brief discussion of volume averaging, we present the exact volume-averaged continuity, momentum, Reynolds stress, turbulent mass-flux, and density-specific volume covariance equations. These equations are identical to those first derived in [1] by the process of ensemble averaging, except here there are explicit integral effects accounting for the variation of flow properties at the time-dependent boundaries of the dispersed phase. The resulting equations will be valid for compressible variable density turbulence with a distinct dispersed phase. Additional physics (evaporation, interfacial deformation) are explicit in the integral expressions. The additional immiscible phase effects represent a perturbation on mean-flow quantities. In the limit of zero-volume fraction of the dispersed phase, the BHR-equations are recovered. The integral expressions owing to the dispersed phase represent additional un-closed terms in the mean-flow equations. Our discussion of these terms is centered around classifying the new terms based on their expected contribution to the flow dynamics. That is, we attempt to classify the salient physics for each new term and present hypotheses as to their respective contribution to the carrier phase (production, transport, dissipation, etc.) Based on the hypothesized effect, we suggest simple models to close the new terms. In our discussion, the terms “particles” and “dispersed phase elements” will be used interchangeably. Additional physics (mass-transfer, interfacial deformation) will be presented as a superimposed effect built into the predominant effect of the dispersed phase on the carrier. The carrier-dispersed phase coupling then will be owing to satisfaction of appropriate boundary conditions at the interface between the respective materials.

Discussion

Review of Volume Averaging

Before extending the BHR equations to include immiscible effects, we briefly review the method of volume averaging to understand how dispersed phased effects will enter the governing equations. We begin by considering the arbitrary control volume containing multiple materials shown in Figure 1. Here, the blue stripes represent the carrier phase which may have a spatially

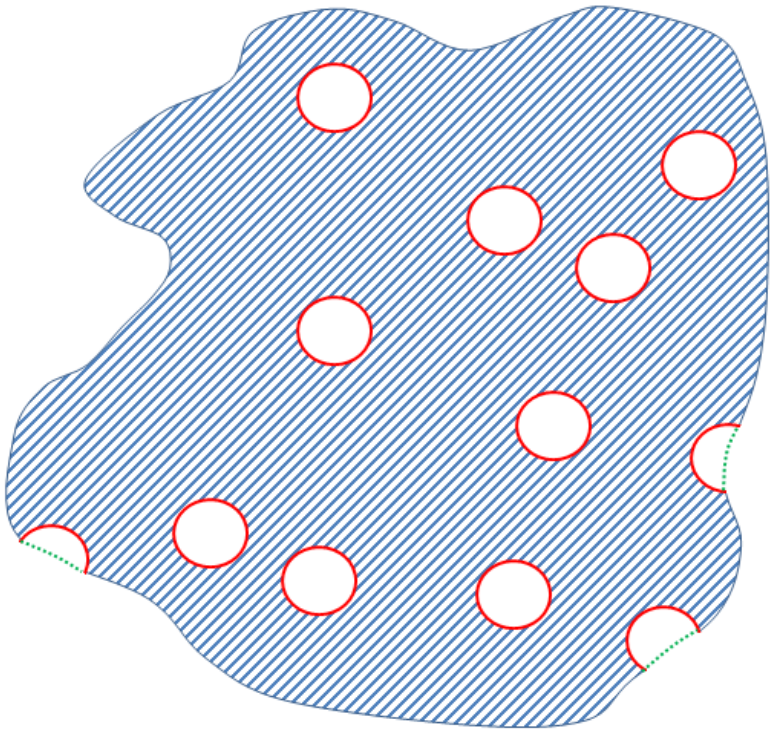


Figure 1: Arbitrary control volume containing carrier phase (stripes) and dispersed phase (circles/semi-circles).

varying density field owing to compressible effects or problem specification. It is important to note here that the carrier-phase is treated as *miscible* so that any immiscible effects are treated as a separate phase with respect to the carrier fluid. The circular (spherical in 3D) elements represent the dispersed phase where the carrier-dispersed phase boundary is represented in red. It should be noted there is no loss in generality in assuming the dispersed phase is made up of identical spherical particles. So long as the *implemented* model is able to distinguish between members of the dispersed phase, the following analysis is valid for dispersed elements of

arbitrary shapes and material properties. Having established an open control volume containing multiple materials, we define the average of some arbitrary flow quantity B[1] in (1) viz. [3]:

$$\bar{B} = \frac{1}{V}\int_{V_c} BdV = \frac{V_c}{V}\frac{1}{V_c}\int_{V_c} BdV = \alpha_c < B > \tag{1}$$

Here, $\bar{B}$ is read as the volume or bulk average of B, V is the total volume contained within the boundaries of the control volume, V_c is the total volume contained within the control volume specified in Figure 1 that is occupied exclusively by carrier-phase material, α_c is the carrier-phase volume fraction, and $< B >$ is read as the carrier-phase, or simply the phase average of B. Having defined a volume average, it is now possible to decompose a flow quantity B in terms of a mean and a fluctuating part viz.

$$B = \bar{B} + B' = \alpha_c < B > + B' \tag{2}$$

Here, B' is the read as the fluctuation of B. By construction, the volume average of the fluctuating term in (2) is zero, viz.:

$$\overline{B'} = \frac{1}{V}\int_{V_c} B'dV = 0 \tag{3}$$

As discussed in [1], it is often desirable to express mean quantities weighted by the local carrier-phase density. In doing so, we may define an alternate volume average called the Favre average [3]:

$$\tilde{B} = \frac{1}{\bar{\rho}_c V}\int_{V_c} \rho_c BdV \tag{4}$$

Here, $\tilde{B}$ is referred as the Favre, mass, or density-weighted average of B and ρ_c is the carrier-phase density. The Favre average allows us to define an alternate decomposition for B:

$$B = \tilde{B} + B'' \tag{5}$$

[1] B may be a scalar, vector, or tensor quantity.

As a consequence, the Favre average of the fluctuating term in (4) is zero, that is:

$$\tilde{B} = \frac{1}{\bar{\rho}_c V}\int_{V_c} \rho_c B'' dV = 0 \tag{6}$$

Denoting the instantaneous carrier-phase velocity as u_i, the turbulent mass-flux $a_i = \overline{u_i''}$ relates respective mean definitions of *velocity*[2] to each other and independently, the respective velocity fluctuation definitions viz.:

$$\tilde{u}_i = \bar{u}_i + a_i \tag{7}$$

$$u_i'' = u_i' - a_i \tag{8}$$

The bulk (1) and Favre average (4) notation will be used throughout where preference for one notation over the other is given based on convenience noting that it is possible to switch between notations by equation (2) and (5). Having defined a volume average, it remains to specify how this average will commute with derivatives. This commutation rule will become necessary once the averaging procedure is applied to the carrier-phase equations of motion. The commutative rule for spatial and temporal derivatives are given in (9) and (10) respectively:

$$\overline{\frac{\partial B}{\partial x_i}} = \frac{\partial \bar{B}}{\partial x_i} - \frac{1}{V}\int_S B n_i dS \tag{9}$$

$$\overline{\frac{\partial B}{\partial t}} = \frac{\partial \bar{B}}{\partial t} + \frac{1}{V}\int_S B(\mathrm{v}_i n_i + \dot{r}) dS \tag{10}$$

where the integral terms in (9) and (10) are consequences of Leibnitz's integral theorem [3].(The reader is referred to Crowe [3] for more details on the derivation of (9) and (10).) The integrals in (9) and (10) are taken over the total surface boundary between dispersed and continuous phases as shown in Figure 1. Therefore, these integrals are understood as the summation of surface integrals over all dispersed phase elements. In essence, the commutative rule in (9) and (10) allows the commutation of derivative and averaging operators at the cost of accounting for the variation of B on the along the dispersed phase boundary, *and* the fact that the dispersed phase boundaries are time dependent as the particles may move around within, enter, or exit the

[2] That is, (7) and (8) hold for the velocity means and fluctuations, but *not* for an arbitrary flow quantity B.

control volume. However, the boundary of the control volume itself is fixed and therefore makes no additional integral contribution in (9) and (10). Therefore, red S in (9) and (10), which refers to the red[3] surfaces in Figure 1 emphasizes that the only dispersed phase surfaces or fractions thereof within the *interior* of the control volume contribute to the surface integrals. Here, v_i is the velocity of the center of mass of any given particle and $\dot{r}$ is the radius rate-of-change of the particle surface w.r.t. to its center of mass. While variation in B along dispersed phase surfaces allows (9) to be non-zero, (10) may be non-zero when B is constant along the surface since $\dot{r}$ need not be constant on S. Note, v_i is constant on S by definition.

We may state this observation more generally and concisely: the integral of *any* mean flow quantity along a dispersed phase surface is zero[4] viz.:

$$\int_S \bar{B} n_i dS = \int_S \tilde{B} n_i dS = 0 \tag{11}$$

The veracity of (11) comes from the definition of the volume average. Upon applying the averaging operator to a flow quantity, the averaged quantity becomes independent of the volume of integration. On the closure of the region of integration, the averaged quantity is invariant so that the averaged quantity must therefore be invariant over each dispersed phase surface. Spatial variations in mean quantities are then understood as varying from one control volume to the next, in the limit as the control volume is reduced to a size much smaller than the average length-scale over which macroscopic flow quantities change. (11) leads to the additional conclusion that immiscible effects are owing to fluctuations[5] viz.:

$$\begin{aligned} \int_S B n_i dS &= \int_S (\bar{B} + B') n_i dS = \int_S B' n_i dS \\ &= \int_S \left(\tilde{B} + B''\right) n_i dS = \int_S B'' n_i dS \end{aligned} \tag{12}$$

This concludes the necessary discussion of volume averaging. In the next section, we present the results of applying volume averaging to the flow equations to yield the BHR equations with explicit integral effects owing to the dispersed phase. We conclude this section by providing a

[3] Red coloring is added to the S in (9) and (10) for emphasis. Following, the occurrence of S in a surface integral has the same meaning regardless of color.

[4] The integration must be performed over a closed surface contour and the surface unit normal must be analytic [15]. This will be the case for simple smooth surfaces such as spheres. As long as the dispersed phase elements are small compared with all flow scales, then all dispersed phase elements will lie entirely within the control volume and each surface integral will be over a closed path.

[5] Mean flow quantities may be present in dispersed phase integrals, but only when combined with other fluctuating quantity(s).

list of useful identities which are used throughout the discussion and Appendix. For any carrier phase quantities A and B we have:

$$\overline{B'} = \overline{\rho_c B''} = 0 \qquad (a)$$

$$\overline{AB} = \bar{A}\bar{B} + \overline{A'B'} \qquad (b)$$

$$\overline{A'B} = \overline{A'B'} = \overline{A'B''} \qquad (c)$$

$$\overline{A''} = -\overline{\rho' A'}/\bar{\rho}_c \qquad (d)$$

$$\overline{\frac{\partial \bar{A}}{\partial x_\iota} B''} = \frac{\partial \bar{A}}{\partial x_i}\overline{B''} \qquad (e)$$

$$\overline{\frac{\partial \tilde{A}}{\partial x_\iota} B''} = \frac{\partial \tilde{A}}{\partial x_i}\overline{B''} \qquad (f)$$

$$\overline{\frac{\partial \tilde{A}}{\partial x_\iota} B'} = \overline{\frac{\partial \bar{A}}{\partial x_\iota} B'} = 0 \qquad (g)$$

Where (a)-(d) are given in [1], and (e)-(g) were found during the present investigation.

BHR Equations with Immiscible Effects

We now present the exact volume-averaged equations for continuity, momentum, Reynolds stress, turbulent mass-flux, and density-specific volume covariance. The details regarding the derivation of these equations can be found in the Appendix. We now present each of these equations individually and discuss their salient aspects. In each of these equations, we have colored dispersed-phase effects in red to emphasize the terms that differ from the original BHR-equations. In the following equations, we will observe that as the continuous phase volume fraction α_c goes to unity, surface elements will shrink to null so that all dispersed phase effects will vanish. In addition, all phase-averages will become volume-averages since the carrier-phase will occupy an entire control volume. Each of the resulting equations will then be identical to those derived by Besnard et al. [1].

Volume-Averaged Continuity Equation with Immiscible Effects

The volume-averaged continuity equation is given as (13):

$$\frac{\partial}{\partial t}[\alpha_c < \rho_c >] + \frac{\partial}{\partial x_i}[\alpha_c < \rho_c > \tilde{u}_i] = \frac{1}{V}\int_S \rho_c w n_i n_i dS = \chi \tag{13}$$

(13) agrees with the expression derived in Crowe [3]. Evidently, the volume-averaged continuity equation differs from the BHR-equation by a single term in the form of a mass source denoted as χ. The mass source depends only on the carrier-phase density evaluated along the surface[6] of each particle as well as the mass-transfer velocity w which is owing to condensation or evaporation effects. Interestingly, the motion of particles into, out of, and within the control volume as well as their respective deformations are irrelevant at least with regard to the averaged continuity equation. Surely the instantaneous carrier-phase velocity within the control volume *is* affected by particles since fluid must move out of the way to accommodate particle flux into new regions where they were not previously. It is with this understanding that (13) represents a series of cancelling effects. That is, in some regions the particle velocity divergence will be positive and other regions it will be negative. So too will the deformation of dispersed phase surfaces contribute to instantaneous continuity. However, the bulk effect of particle movement and deformation averaged over the control volume makes no contribution to average continuity. Then, only *net* sources of mass between the dispersed and carrier-phase may affect the averaged continuity equation. The term net is used here to emphasize competing effects, that is, nothing in χ suggests that this term must be a source and in practice it may be a sink of mass from the

[6] Since the carrier-phase density may be ill-defined at material boundaries, the density here is understood as a limit taking into account carrier-phase density in the neighborhood of the boundary.

carrier-phase. The net effect of χ on any given control volume can be found by performing the integration in (13) on each particle surface, or fraction there-of within a given control volume and then summing these contributions. Such an integral evaluation would likely be done using a numerical integration procedure. This procedure would require knowledge of the geometry of each dispersed surface (which may vary from one particle to the next in poly-disperse flows) and a method for determining the mass-transfer rate at a sample of points along each respective dispersed phase surface. When the details of S are complicated or when the calculation of w on S is impractical, a simple model to account for the bulk effect of the mass-source may be preferred. Crowe [3] proposes a simple model for the mass source in the averaged continuity equation by removing details of the variation of in mass-transfer rate at the surface:

$$\chi_{model} = -\frac{1}{V}\sum_{k}\dot{m}_k \tag{14}$$

The summation in (14) is performed over the k-particles contained at least partially within a given control volume, and $\dot{m}_k$ is the mass rate-of-change of the k^{th} particle. If each particle is small compared to the characteristic flow dimensions, the details of each particle's surface may be un-important and it may be sufficient to have the same model for $\dot{m}$ for each particle. Such a model would be dependent only on the local carrier-phase density, and perhaps on the material properties of the particle itself.

Volume-Averaged Momentum Equation with Immiscible Effects

The volume-averaged momentum equation is presented in (15):

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > \tilde{u}_i + \frac{\partial}{\partial x_j}\alpha_c < \rho_c > \tilde{u}_i\tilde{u}_j =$$

$$-\frac{\partial}{\partial x_j}\alpha_c < \rho_c > \tilde{R}_{ij} - \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \overline{\tau_{ij}}}{\partial x_j} + \alpha_c < \rho_c > g_i + C_i \tag{15}$$

The expression for the volume-averaged momentum equation (15) agrees with [3]. Here, p is the pressure, τ_{ij} is the deviatoric stress tensor, $\tilde{R}_{ij} = \overline{\rho_c u_i'' u_j''}/\bar{\rho}_c$ is the generalized Reynolds stress which shall henceforth be referred to as the Reynolds stress, g_i is the gravitational acceleration, and C_i is the total contribution of dispersed phase effects on the mean momentum equation and is given in (16):

$$C_i = \frac{1}{V}\int_S \left(\rho_c u_i w + p n_i - \tau_{ij} n_j\right) dS \tag{16}$$

The dispersed phase effects in (16) may be categorized into two groups: the first term in the integrand of (16) is owing to mass-transfer effects between the dispersed and carrier phase while the remaining terms are owing to variation in stress exerted over each dispersed phase surface. C_i may therefore be decomposed into a thrust contribution owing to mass-transfer, and a drag contribution owing to stress imbalance. Of course, lift components may be present in the latter contribution; the term drag is used with the understanding that the net effect of a particle on the carrier phase is that of flow impedance, and the sense of the resulting drag will likely have some component orthogonal to a particle's instantaneous velocity vector. Therefore, upon decomposition of C_i into thrust and drag effects, we have without approximation:

$$C_i = F_{thrust} + F_{drag} \tag{17}$$

If each particle is non-deformable (i.e. $\dot{r}$ is zero[7]), then the thrust term may be modeled viz. [3]:

$$F_{thrust} \approx F_{t,model} = -\frac{1}{V}\sum_k \mathrm{v}_{i,k}\dot{m}_k \tag{18}$$

Such a model is expected to be a reasonable approximation when the dispersed phase elements are small compared with flow scales, so that the details of each particle's geometry are unimportant. Similarly, if the dispersed phase elements are much smaller than relevant flow scales, specifically the turbulent Kolmogorov scales [11], the drag term in (17) may be modeled via a relationship similar to that derived by Maxey and Riley [7] viz.:

$$F_{drag} \approx F_{d,model} = F_{grav} + F_{visc,st} + F_{hist} + F_{added} + F_{p,visc} \tag{19}$$

[7] Surely $\dot{r}$ will be coupled to mass transfer effects since gain or loss of mass over long enough times will result in a measurable change in the volume of a dispersed phase element, and as a consequence, the position of its surface w.r.t. to its center of mass. Therefore, we must assume the timescale over which r changes is much greater than other relevant timescales (mass-transfer and flow timescales). When this assumption fails, an additional term in the summation in (18) of the form $\dot{r}_k\dot{m}_k$ will become necessary.

Where F_{grav} is the gravitational force, $F_{visc,st}$ is the steady Stokes drag which may be modified for finite Reynolds number effects [11], and the remaining terms are unsteady effects due to history, added mass, and combined pressure gradient and viscous stresses respectfully. When the particle Reynolds number is small, the Stokes term will be the primary contribution to the drag and the remaining terms will be small except possibly at small times [11]. However, when the Mach number is not small so that compressibility effects are important or when particles are not small compared to the smallest turbulent scales, the modeled drag equation would require additional treatment. Additionally, deformable particles have the ability to store strain-energy at their boundary with the carrier-phase. The model for the pressure and viscous terms in (16) would then also require treatment of interfacial tension effects using [12] for example.

Reynolds Stress Equation with Immiscible Effects

The exact equation for the Reynolds stresses, including immiscible effects is given in (20):

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > \tilde{R}_{ij} + \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{u}_k \tilde{R}_{ij} = \tag{20}$$

$$\underbrace{-< \rho_c > \tilde{R}_{jk}\frac{\partial \tilde{u}_i}{\partial x_k} - \alpha_c < \rho_c > \tilde{R}_{ik}\frac{\partial \tilde{u}_j}{\partial x_k} + a_i\frac{\partial \bar{p}}{\partial x_j} + a_j\frac{\partial \bar{p}}{\partial x_i} - a_i\frac{\partial \tilde{\tau}_{jk}}{\partial x_k} - a_j\frac{\partial \tilde{\tau}_{ik}}{\partial x_k}}_{\text{Production}}$$

$$\text{Transport}\left\{\begin{array}{l} -\frac{\partial}{\partial x_k}\alpha_c < \rho_c u_i'' u_j'' u_k'' > -\alpha_c - \frac{\partial}{\partial x_j}\alpha_c < p' u_i'' > - \frac{\partial}{\partial x_i}\alpha_c < p' u_j'' > \\ \qquad + \frac{\partial}{\partial x_k}\alpha_c < u_i'' \tau_{jk}'' > + \frac{\partial}{\partial x_k}\alpha_c < u_j'' \tau_{ik}'' > \end{array}\right.$$

$$\underbrace{+\alpha_c < p'\frac{\partial u_j''}{\partial x_i} > + \alpha_c < p'\frac{\partial u_i''}{\partial x_j} >}_{\text{Pressure-Strain}} \quad \underbrace{-\alpha_c < \tau_{jk}''\frac{\partial u_i''}{\partial x_k} > - \alpha_c < \tau_{ik}''\frac{\partial u_j''}{\partial x_k} >}_{\text{Dissipation}}$$

$$\underbrace{+ E_{ij} + P_{ij} - T_{ij}}_{\text{Dispersed Phase Effects}}$$

and,

$$E_{ij} = \frac{1}{V}\int_S \left[\rho_c w\left(u_i'' u_j'' - \tilde{u}_i \tilde{u}_j\right)\right] dS \tag{21}$$

$$P_{ij} = \frac{1}{V}\int_S p'\left(u_i'' n_j + u_j'' n_i\right) dS \tag{22}$$

$$T_{ij} = \frac{1}{V}\int_S \left(u_i'' \tau_{jk}'' + u_j'' \tau_{ik}''\right) n_k \, dS \tag{23}$$

If the density of the carrier fluid is taken to be constant and no mass-transfer takes place at surfaces separating the carrier and dispersed phases, the Reynolds stress equation (20) agrees with the expression derived in [3]. In the absence of immiscible effects all-together, (20) will reduce to the expression originally derived in [1]. The unclosed production, transport, and pressure-strain terms in (20) may be closed using direct models discussed in [2]. The turbulent mass-flux, a_i, appearing as a coefficient on the spatial derivatives of mean pressure and viscous stress is also unclosed; its closure comes via a transport equation discussed in the next section. The dissipation term in (20) may be closed using a compressible analog to the dissipation equation derived in [14]. Then, as with the momentum equation, the dispersed phase contribution to the Reynolds stress may be categorized as owing to mass-transfer effects (21) and stress effects (22), (23). (22) and (23) are significant since they establish that rigid, non-evaporating particles may still contribute as a source of Reynolds stress. The mass-transfer term (21) may be re-written in a more convenient form for modeling. Without approximation, we have:

$$E_{ij} = \frac{1}{V}\int_S \left[\rho_c w\left(u_i'' u_j'' - \tilde{u}_i \tilde{u}_j\right)\right] dS$$

$$= \frac{1}{V}\int_S \left[\rho_c w\left(u_i u_j - \tilde{u}_i u_j - \tilde{u}_j u_i + \tilde{u}_i \tilde{u}_j - \tilde{u}_i \tilde{u}_j\right)\right] dS$$

$$= \frac{1}{V}\int_S \rho_c w u_i u_j \, dS - \tilde{u}_i \frac{1}{V}\int_S \rho_c w u_j \, dS - \tilde{u}_j \frac{1}{V}\int_S \rho_c w u_i \, dS$$

$$= \frac{1}{V}\int_S \rho_c w\{\mathrm{v}_i + (\dot{r} + w) n_i\}\{\mathrm{v}_j + (\dot{r} + w) n_j\} \, dS$$

$$- \tilde{u}_i \frac{1}{V}\int_S \rho_c w\{\mathrm{v}_j + (\dot{r} + w) n_j\} \, dS - \tilde{u}_j \frac{1}{V}\int_S \rho_c w\{\mathrm{v}_i + (\dot{r} + w) n_i\} \, dS$$

$$= \frac{1}{V}\int_S \rho_c w\{\mathrm{v}_i\mathrm{v}_j + \mathrm{v}_i(\dot{r}+w)n_j + \mathrm{v}_j(\dot{r}+w)n_i + (\dot{r}+w)^2\delta_{ij}\}\,dS$$

$$- \tilde{u}_i\,\frac{1}{V}\int_S \rho_c w\{\mathrm{v}_j + (\dot{r}+w)n_j\}\,dS - \tilde{u}_j\frac{1}{V}\int_S \rho_c w\{\mathrm{v}_i + (\dot{r}+w)n_i\}\,dS \quad (24)$$

If $\dot{r}$ is taken to be zero, (24) may be expressed in modeled form using expressions analogous to (14) and (18):

$$E_{ij} \approx E_{ij,model} =$$

$$-\frac{1}{V}\sum_k \mathrm{v}_{i,k}\mathrm{v}_{j,k}\dot{m}_k - \frac{1}{V}\sum_k \frac{\mathrm{v}_{i,k}(\dot{m}_k)^2 n_j}{\rho_{c,k}A_k} - \frac{1}{V}\sum_k \frac{\mathrm{v}_{j,k}(\dot{m}_k)^2 n_i}{\rho_{c,k}A_k} - \frac{1}{V}\sum_k \frac{(\dot{m}_k)^3}{(\rho_{c,k})^2(A_k)^2}\delta_{ij}$$

$$-\frac{1}{V}\sum_k \tilde{u}_i\mathrm{v}_{j,k}\dot{m}_k - \frac{1}{V}\sum_k \frac{\tilde{u}_i(\dot{m}_k)^2 n_j}{\rho_{c,k}A_k} - \frac{1}{V}\sum_k \tilde{u}_j\mathrm{v}_{i,k}\dot{m}_k - \frac{1}{V}\sum_k \frac{\tilde{u}_j(\dot{m}_k)^2 n_i}{\rho_{c,k}A_k} \quad (25)$$

Here, $\rho_{c,k}$ and A_k are respectively the carrier phase density and surface area at the k^{th} particle. The most prominent feature of (25) is that the dispersed phase source term owing to mass-transfer has a cubic dependence on the mass-transfer rate $\dot{m}$. If the mass-transfer rate is not small, this source term may have a substantial contribution to production. Further, inspection of (24) reveals the cubic contribution is directly to the TKE while the linear and quadratic mass-transfer terms affect both the diagonal and off-diagonal components of the Reynolds stress. It should be noted that (25) requires only an external model for $\dot{m}_k$ since $\tilde{u}_i$ is closed and $\mathrm{v}_{i,k}$ may be accounted for via Lagrangian particle tracking.

We now turn our attention to modeling the pressure and viscous work terms in (22) and (23). Both of these equations depend on the product of two fluctuating quantities and should yield similar modeled forms. Examining the first term in the integrand of (22), we have:

$$\frac{1}{V}\int_S p'u_i''n_j\,dS = \frac{1}{V}\int_S p'(u_i - \tilde{u}_i)n_j\,dS = \frac{1}{V}\int_S p'u_i n_j\,dS - \tilde{u}_i\frac{1}{V}\int_S p'n_j\,dS$$

$$= \frac{1}{V}\int_S p'\{\mathrm{v}_i + (\dot{r}+w)n_i\}\,n_j dS - \tilde{u}_i\frac{1}{V}\int_S p'n_j\,dS \quad (26a)$$

Taking $\dot{r} = 0$, (26a) may be expressed as [3]:

$$\approx -\frac{1}{V}\sum_k \mathrm{v}_{i,k} F^p_{j,k} - \frac{1}{V}\sum_k \frac{F^p_{j,k}\dot{m}_k n_i}{\rho_{c,k} A_k} - \tilde{u}_i \frac{1}{V}\sum_k F^p_{j,k} \tag{26b}$$

Here, $F^p_{j,k}$ is the net fluid pressure force in the j^{th} direction on the k^{th} particle. The remaining term in (22) as well as the terms in (23) may be cast into an analogous form so that the total contribution by pressure (22) and viscous work (23) interaction with the dispersed phase may be modeled as (27):

$$-\frac{1}{V}\sum_k \mathrm{v}_{i,k} F_{j,k} - \frac{1}{V}\sum_k \frac{F_{j,k}\dot{m}_k n_i}{\rho_{c,k} A_k} - \tilde{u}_i \frac{1}{V}\sum_k F_{j,k} - \frac{1}{V}\sum_k \mathrm{v}_{j,k} F_{i,k} - \frac{1}{V}\sum_k \frac{F_{i,k}\dot{m}_k n_j}{\rho_{c,k} A_k} - \tilde{u}_j \frac{1}{V}\sum_k F_{i,k} \tag{27}$$

In this case, $F_{j,k}$ is interpreted as the net fluid force in the j^{th} direction on the k^{th} particle owing to pressure *and* viscous stresses. Therefore, a suitable expression for $F_{j,k}$ may be given by (19). All other terms in (27) are either closed or may be determined through methods discussed previously. We stress there is no loss of generality in neglecting the $\dot{r}$ term. In flows where rapid variation of the interface between the dispersed and carrier phase may exist, such as in bubbly flows, $\dot{r}$ may be a significant term. Additional terms of the form $\dot{r}_k$ will then be present in (27) as well as (18) and (25). So long as a reasonable model exists for $\dot{r}$, these expressions for dispersed phase effects will still be considered closed.

Turbulent Mass Flux Equation with Immiscible Effects

We now present the transport equation for the turbulent mass flux or density-velocity correlation, a_i, with explicit integral terms accounted for immiscible dispersed phase effects (28):

$$\frac{\partial \overline{\rho_c} a_i}{\partial t} + \frac{\partial \overline{\rho_c} \tilde{u}_j a_i}{\partial x_j}$$
$$= -b\frac{\partial \overline{\sigma_{ij}}}{\partial x_i} - \tilde{R}_{ij}\frac{\partial \overline{\rho_c}}{\partial x_j} + \overline{\rho_c}\frac{\partial}{\partial x_j} a_i a_j - \overline{\rho_c} a_j \frac{\partial \overline{u}_i}{\partial x_j} - \overline{\rho_c}\frac{\partial}{\partial x_j}\left(\frac{\overline{\rho' u_i' u_j'}}{\overline{\rho_c}}\right) - \overline{\rho_c}\overline{\left(\frac{1}{\rho_c}\right)' \frac{\partial \sigma_{ij}'}{\partial x_i}}$$

$$-\overline{\rho_c}\overline{u_i'\frac{\partial u_j''}{\partial x_j}} - B_i + a_i\chi \tag{28}$$

Where B_i is given as (29):

$$B_i = \overline{(\tilde{u}_i \cdot \chi - C_i)} + \overline{\rho_c}\frac{1}{V}\int_S u_i'' w n_j n_j dS - (b+1)\frac{1}{V}\int_S \sigma_{ij}' n_j dS \tag{29}$$

Neglecting immiscible effects, (28) agrees with the miscible a_i equation first derived in [1] except for an additional dilatation term which was assumed to be small and therefore neglected in the original derivation. Models for the unclosed triple-product and specific volume-stress divergence terms in (28) are discussed in [2]. The $a_i\chi$ term in (28) may be modeled using (14) scaled by a_i. The first term in (29) is the volume average of a purely mass-transfer effect, $\tilde{u}_i \cdot \chi$, and a term involving both mass-transfer as well as pressure and viscous effects, C_i, as shown in (16). This term is a natural consequence of the derivation procedure discussed in the Appendix. However, referring to the definition of the volume average in (1), the region of volume integration is disjoint from the region of surface integration except at the surface of each particle. We will not dwell on this point more except to say that interpretation of this term is ongoing and that it has not been formally proved whether the volume integral of surface integral effects will be zero by Lebesgue arguments or will have an effect analogous to averaging a dispersed phase effect over an entire control volume. A similar term will be found in the b equation, which will also be a natural consequence of its derivation. This term will also require further investigation before it can be formulated as a proper model. Using similar arguments as those used in modeling the immiscible effects in the previous transport equations, the remaining terms in (29) may be expressed as (30):

$$-\frac{1}{V}\sum_k \mathrm{v}_{i,k}\dot{m}_k - \frac{1}{V}\sum_k \frac{(\dot{m}_k)^2 n_i}{\rho_{c,k}A_k} - \tilde{u}_i\frac{1}{V}\sum_k \dot{m}_k + (b+1)\frac{1}{V}\sum_k F_{i,k} \tag{30}$$

In (30), $b = -\overline{\rho' v'}$ is the density-specific volume covariance, the particle velocity $\mathrm{v}_{i,k}$ may be found through particle-tracking, $F_{i,k}$ is the net combination of pressure and viscous drag on the k^{th} particle and may be modeled using (19), $\dot{m}_k$ is the mass-transfer rate of the k^{th} particle and may be found via an appropriate model, and $\dot{r}_k$ terms have been neglected under the assumptions

previously discussed. Further investigation will be needed to understand and ultimately close the first term in (29) which will result in a fully closed model for the immiscible effects in the a_i equation.

Density-Specific Volume Covariance Equation with Immiscible Effects

Finally, we present the transport equation for the density-specific volume covariance b, including explicit and exact representation of the immiscible dispersed phase effects (31):

$$\frac{\partial b}{\partial t} + \bar{u}_i \frac{\partial b}{\partial x_i} + \left(\frac{b+1}{\bar{\rho}_c}\right) \frac{\partial}{\partial x_i} \overline{\rho_c} a_i + \overline{\rho_c} \frac{\partial}{\partial x_i} \overline{v' u_i{}'} - 2\overline{\rho_c} \overline{v' \frac{\partial u_i'}{\partial x_i}} = H \tag{31}$$

Where the sum of the immiscible effects H is expressed as (32):

$$H = -\frac{1}{V}\int_S \rho_c' v w n_i n_i dS + \overline{v\chi} - (2b+1)\frac{1}{V}\int_S u_i' n_i dS \tag{32}$$

In the limit of fully miscible turbulence, the immiscible effects in (31) are neglected and the resulting b equation is exactly the expression originally derived in [1]. Models for the specific volume-velocity correlation and dilatation terms in (31) are discussed in [2].

We now examine (32) to propose suitable models for the immiscible effects in the b equation (31). The middle term in (32) exists if and only if the particles experience mass-transfer, as is evident in the expression for χ (13). However, this term involves the volume average of a surface integral effect. As discussed in the context of the a_i equation, interpreting this term requires further investigation. The first term in (32) will be non-zero only when there is mass transfer between the dispersed and carrier-phase. We express this term in a more convenient form for modeling (33):

$$\frac{1}{V}\int_S \rho_c' v w dS = \frac{1}{V}\int_S (\rho_c - \bar{\rho}_c) v w dS = \frac{1}{V}\int_S (1 - \bar{\rho}_c v) w dS \tag{33}$$

Therefore, an appropriate model for (33) may be expressed in the form (34):

$$\frac{1}{V}\int_S (1-\bar{\rho}_c v) w dS \approx -\frac{1}{V}\sum_k \dot{m}_k /\rho_{c,k} - \frac{1}{V}\sum_k \dot{m}_k /\bar{\rho}_{c,k} \tag{34}$$

Here, $\rho_{c,k}$ and $\bar{\rho}_{c,k}$ are respectively the instantaneous and average carrier-phase density evaluated at the k^{th} particle location.

Finally, we may re-write the final term in (32) viz.:

$$(2b+1)\frac{1}{V}\int_S u_i' n_i dS = (2b+1)\frac{1}{V}\int_S u_i n_i dS$$

$$= (2b+1)\frac{1}{V}\int_S \{\mathrm{v}_i + (\dot{r}+w) n_i\} n_i dS \tag{35}$$

The v_i term is constant on S and therefore makes no contribution to the integral in (35). Taking into account the other immiscible effects in (32), we arrive at the following observation: if the $\dot{r}$ term in (35) is neglected, then only mass-transfer between the dispersed and carrier phase contributes to a source of b. In the limit of rigid, non-evaporating (or condensing) particles, the immiscible b equation is identical to the miscible b equation. For completeness, a potential model for (35) is given in (36):

$$(2b+1)\frac{1}{V}\int_S \{\mathrm{v}_i + (\dot{r}+w) n_i\} n_i dS \approx -(2b+1)\left\{\frac{1}{V}\sum_k \dot{r}_k A_k + \frac{1}{V}\sum_k \dot{m}_k /\rho_{c,k}\right\} \tag{36}$$

Here, $\dot{r}_k$ and A_k are respectively the radius rate-of-change surface area of the k^{th} particle. Upon developing a suitable model for the $\overline{v\chi}$ in (32), the immiscible effects in the b equation may be considered closed.

This concludes our presentation of the mass, momentum, Reynolds stress, turbulent mass flux, and density-specific volume covariance equations with explicit integral terms owing to immiscible effects. More details on the derivation of these respective equations can be found in the Appendix.

Conclusions and Future Work

In this report, we extended the work of Besnard et al. [1] and Crowe [3] by deriving explicit integral source terms owing to immiscible effects for the volume-averaged continuity and momentum equations, as well as the Reynolds stress, turbulent mass-flux, and density-specific volume covariance transport equations. We then proposed models to close the integral terms using expressions analogous to those discussed in [3]. Incorporating models for the immiscible effects along with models [2] developed for the carrier-phase un-closed turbulent quantities, the resulting set of coupled non-linear PDEs forms a closed set of equations that may be simulated. These equations are valid for compressible multiphase turbulent flows where a definable interface separates the primary carrier phase fluid with discrete pieces of dispersed phase material (particles, droplets, bubbles). The models for the immiscible terms and the method of volume averaging itself are valid when the pieces of dispersed phase material are small in comparison with relevant flow scales. Specifically, the drag term proposed by Maxey and Riley [7] which is used in several of the model equations should be valid provided the dispersed phase is made up of rigid particles much smaller than the Kolmogorov scales [11]. These extended BHR equations are not expected to be valid when the dispersed phase volume fraction becomes large or when the dispersed phase elements become comparable to the smallest flow scales. In the case of deformable droplets or bubbles, a model for $\dot{r}$ becomes necessary. When the flow involves two phases of the same molecular component—water liquid and vapor for example—mass transfer effects may exist which require modeling.

Following Crowe's work [3], we have shown the only contribution to mean continuity takes the form of a mass-source while mean momentum involves both mass-transfer as well as pressure and viscous stress effects. The immiscible integrals in the Reynolds stress and turbulent mass-flux equations may be decomposed into sources involving only mass-transfer or stress effects. If the dispersed phase is comprised of evaporating or condensing material whose surface changes over much longer time scales than the characteristic rate of mass-transfer (i.e. $\dot{r} \ll w$), then only mass-transfer effects contribute to sources of density-specific volume covariance b.

Two immiscible terms, one in the a_i equation and one in the b equation involve the volume averages of flow quantities integrated over dispersed phase surfaces. These barred-terms are composed of classifiable effects including mass-transfer and stress effects, but the terms as a whole require further investigation before they can be modeled. In the present work, there was insufficient time to derive the immiscible effects for the turbulent heat flux equation originally derived in [1]. Future work should also focus on extending the dissipation equation derived by Schwarzkopf et al. [14] for incompressible particle-laden flows to include both the effects of compressibility and mass-transfer. In this report, we have proposed simple models for the immiscible effects resulting from the volume averaging procedure. However, validation of these models is necessary to assess their applicability in various flow regimes.

Acknowledgements

I would like to thank my mentor John Schwarzkopf for tremendously valuable support and feedback. I offer my gratitude to my colleagues Sasha Tan-Torres and Dan Israel for their thoughtful comments. I am grateful to Scott Runnels for organizing the 2013 Computational Physics workshop and inviting me to participate this summer. This work would not be possible without the support allocated to the workshop by the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

References

[1] D. Besnard, F. H. Harlow, R. M. Rauenzahn, C. Zemach, "Turbulence Transport Equations for Variable-Density Turbulence and Their Relationship to Two-Field Models," LA-UR-12303, Los Alamos National Laboratory, Los Alamos, NM, 1992.

[2] J. D. Schwarzkopf, D. Livescu, R. A. Gore, R. M. Rauenazhn, J. R. Ristorcelli (2011): Application of a second-moment closure model to mixing processes involving multicomponent miscible fluids, Journal of Turbulence, 12, N49.

[3] C. Crowe, J.D. Schwarzkopf, M. Sommerfeld, Y. Tsuji, Multiphase Flows with Droplets and Particles, 2nd Ed., CRC Press, 2012.

[4] G. G. Stokes, "On the effect of the internal friction of fluids on the motion of pendulums," Trans. Phil. Soc. Vol. IX. p.8.

[5] A. B. Basset, "On the Motion of a Sphere in a Viscous Liquid," Phil. Trans. R. Soc. A, Vol. 179, (1888), pp. 43-63.

[6] G.K. Batchelor, An Introduction to Fluid Dynamics, Cambridge University Press, 1967.

[7] M.R. Maxey, J.J. Riley, "Equation of motion for a small rigid sphere in a nonuniform flow," Phys. Fluids **26**, 883 (1983); doi: 10.1063/1.864230.

[8] M.R. Maxey (1987). "The gravitational settling of aerosol particles in homogeneous turbulence and random flow fields." Journal of Fluid Mechanics, 174, pp. 441-465 doi:10.1017/S0022112087000193.

[9] K.D. Squires, J. K. Eaton, (1990). "Particle Response and turbulence modification in isotropic turbulence," Phys. Fluids A 2 (7), pp. 1191-1203.

[10] S. Sundaram, L. R. Collins (1999) "A numerical study of modulation of isotropic turbulence by suspended particles," Journal of Fluid Mechanics, 379, pp 105-143.

[11] T.M. Burton, J.K. Eaton, "Fully resolved simulations of particle-turbulence interaction," J. Fluid Mech. (2005) vol. 545, pp.67-111. Doi:10.1017/S0022112005006889.

[12] J. U. Brackbill, D. B. Kothe, C. Zemach, "A Continuum Method for Modeling Surface Tension," J. Comp. Phys. **100**, 335-354 (1992).

[13] R. A. Gore, C. T. Crowe, "Effect of particle size on modulating turbulent intensity," Int. J. Multiphase Flow, Vol. 15, No.2, pp. 279-285, 1989.

[14] J.D. Schwarzkopf, C.T. Crowe, P. Dutta, "A Turbulence Dissipation Model for Particle Laden Flow," AIChE J, Vol. 55, No. 6, 1416-1425, 2009, doi 10.1002/aic.11773.

[15] S.D. Fisher, Complex Variables, 2nd Ed. Dover Publications, 1999.

[16] S. Sundaram, and L. R. Collins (1997). "Collision statistics in an isotropic particle-laden Turbulent suspension. Part 1. Direct numerical simulations." Journal of Fluid Mechanics, 335, pp. 75-109 doi:10.1017/S0022112096004454.

Appendix

Derivation of Filtered Continuity Equation

The continuity equation for carrier phase is written as:

$$\frac{\partial \rho_c}{\partial t} + \frac{\partial}{\partial x_i}\rho_c u_i = 0 \qquad (A.1)$$

Where ρ_c and u_i are respectively the instantaneous carrier phase density and velocity. Applying volume averaging to each of the terms on the left hand side of (A.1), we have:

$$\overline{\frac{\partial \rho_c}{\partial t}} = \frac{\partial \overline{\rho_c}}{\partial t} + \frac{1}{V}\int_S \rho_c(\mathrm{v}_i n_i + \dot{r})dS \qquad (A.2)$$

And,

$$\overline{\frac{\partial}{\partial x_\iota}\rho_c u_\iota} = \frac{\partial \overline{\rho_c u_\iota}}{\partial x_i} - \frac{1}{V}\int_S \rho_c u_i n_i dS \qquad (A.3)$$

Using the definitions (1) and (4), we have:

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > + \frac{\partial}{\partial x_i}\alpha_c < \rho_c > \tilde{u}_i = \frac{1}{V}\int_S \rho_c u_i n_i dV - \frac{1}{V}\int_S \rho_c(\mathrm{v}_i n_i + \dot{r})dS \qquad (A.4)$$

Finally, the carrier phase velocity at the dispersed phase surface can be expressed as $u_i|_S = \mathrm{v}_i + (\dot{r} + w)n_i$ [3], where w is the mass-transfer velocity, so that (A.4) becomes:

$$\boxed{\frac{\partial}{\partial t}\alpha_c < \rho_c > + \frac{\partial}{\partial x_i}\alpha_c < \rho_c > \tilde{u}_i = \frac{1}{V}\int_S \rho_c w dS = \chi} \qquad (A.5)$$

Where χ is a source term accounting for mass transfer between the carrier and dispersed phase owing to evaporation, condensation, or other effects.

Derivation of Filtered Momentum Equation

The momentum equation for the carrier phase is written in (A.6):

$$\frac{\partial}{\partial t}\rho_c u_i + \frac{\partial}{\partial x_j}\rho_c u_i u_j = \frac{\partial}{\partial x_j}\sigma_{ij} + \rho_c g_i \qquad (A.6)$$

Where $\sigma_{ij} = -p\delta_{ij} + \tau_{ij}$, is the fluid stress tensor, p is the pressure, τ_{ij} is the deviatoric stress tensor, and g_i is the gravitational acceleration. Applying volume averaging term-by-term in (A.6), we have:

$$\overline{\frac{\partial}{\partial t}\rho_c u_\iota} = \frac{\partial}{\partial t}\overline{\rho_c u_\iota} + \frac{1}{V}\int_S \rho_c u_i\left(\mathrm{v}_j n_j + \dot{r}\right)dS \qquad (A.7)$$

$$\overline{\frac{\partial}{\partial x_J}\rho_c u_\iota u_J} = \frac{\partial}{\partial x_j}\overline{\rho_c u_\iota u_J} - \frac{1}{V}\int_S \rho_c u_i u_j n_j dS \qquad (A.8)$$

$$\overline{\frac{\partial}{\partial x_J}\sigma_{\iota J}} = \overline{-\frac{\partial p}{\partial x_\iota}} + \overline{\frac{\partial \tau_{\iota J}}{\partial x_J}} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \overline{\tau_{\iota J}}}{\partial x_j} + \frac{1}{V}\int_S \left(pn_i - \tau_{ij}n_j\right)dS \qquad (A.9)$$

$$\overline{\rho_c g_\iota} = \overline{\rho_c} g_i \qquad (A.10)$$

The $\overline{\rho_c u_\iota u_J}$ term in (A.8) requires further treatment. Using the Favre decomposition, this can be expressed using in terms of closed terms, the bulk-average density and Favre average velocity, and the Reynolds stresses viz.

$$\overline{\rho_c u_\iota u_J} = \overline{\rho_c(\tilde{u}_\iota + u_\iota'')\left(\tilde{u}_J + u_J''\right)} = \overline{\rho_c \tilde{u}_\iota \tilde{u}_J} + \overline{\rho_c \tilde{u}_\iota u_J''} + \overline{\rho_c u_\iota'' \tilde{u}_J} + \overline{\rho_c u_\iota'' u_J''} \qquad (A.11)$$

The bulk-averaged velocity $\tilde{u}_i$ is independent of the volume average. By construction, the density-weighted volume average of a fluctuation u_i'' is zero, so that the second and third terms on the right-hand side of (A.11) are zero. We therefore have:

$$\overline{\rho_c u_\iota u_J} = \overline{\rho_c(\tilde{u}_\iota + u_\iota'')\left(\tilde{u}_J + u_J''\right)} = \overline{\rho_c}\tilde{u}_i\tilde{u}_j + R_{ij} \qquad (A.12)$$

Where $R_{ij} = \overline{\rho_c u_\iota'' u_J''}$ is the mass-weighted Reynolds stress. The filtered momentum equation can now be written:

$$\frac{\partial}{\partial t}\overline{\rho_c u_\iota} + \frac{\partial}{\partial x_j}\overline{\rho_c}\tilde{u}_i\tilde{u}_j + \frac{\partial}{\partial x_j}R_{ij} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \overline{\tau_{\iota J}}}{\partial x_j} + \overline{\rho_c}g_i +$$

$$\frac{1}{V}\int_S \left(pn_i - \tau_{ij}n_j\right)dS + \frac{1}{V}\int_S \rho_c u_i u_j n_j dS - \frac{1}{V}\int_S \rho_c u_i\left(\mathrm{v}_j n_j + \dot{r}\right)dS \qquad (A.13)$$

The filtered momentum equation (A.13) may also be expressed as (A.14):

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > \tilde{u}_i + \frac{\partial}{\partial x_j}\alpha_c < \rho_c > \tilde{u}_i\tilde{u}_j = -\frac{\partial}{\partial x_j}\alpha_c < \rho_c > \tilde{R}_{ij} - \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \overline{\tau_{\iota J}}}{\partial x_j} + \alpha_c < \rho_c > g_i + C_i \qquad (A.14)$$

Here, $\boxed{C_i = \frac{1}{V}\int_S \left(\rho_c u_i w + pn_i - \tau_{ij}n_j\right)dS}$ and $\tilde{R}_{ij} = \overline{\rho_c u_\iota'' u_J''}/\bar{\rho}_c$ is the normalized Reynolds stress. The second and third integral terms in (A.13) were reduced via the same procedure employed in re-writing (A.4) as (A.5).

Derivation of Reynolds Stress Equation

The transport equation for the Reynolds stresses is obtained by computing the first cross-moments of the momentum equation and subtracting the corresponding means viz.

$$\{RS\} = \overline{u_\iota \cdot NS_\jmath + u_\jmath \cdot NS_\iota} - \left(\tilde{u}_i \cdot \overline{NS}_j + \tilde{u}_j \cdot \overline{NS}_i\right) \qquad (A.15)$$

Where NS_j is the j^{th} component of the un-filtered momentum equation. Starting with the first two terms in (A.15), we respectfully have:

$$u_i \cdot NS_j = u_i \cdot \left\{\frac{\partial}{\partial t}\rho_c u_j + \frac{\partial}{\partial x_k}\rho_c u_j u_k = \frac{\partial}{\partial x_k}\sigma_{jk} + \rho_c g_j\right\} \qquad (A.16)$$

And,

$$u_j \cdot NS_i = u_j \cdot \left\{\frac{\partial}{\partial t}\rho_c u_i + \frac{\partial}{\partial x_k}\rho_c u_i u_k = \frac{\partial}{\partial x_k}\sigma_{ik} + \rho_c g_i\right\} \qquad (A.17)$$

Using the product rule of differentiation, and writing out the stress tensor explicitly, the sum of (A.16) and (A.17) can be expressed as (A.18):

$$\frac{\partial}{\partial t}\rho_c u_i u_j + \frac{\partial}{\partial x_k}\rho_c u_i u_j u_k = -u_i \frac{\partial p}{\partial x_j} - u_j \frac{\partial p}{\partial x_i} + u_i \frac{\partial \tau_{jk}}{\partial x_k} + u_j \frac{\partial \tau_{ik}}{\partial x_k} + \rho_c u_i g_j + \rho_c u_j g_i \qquad (A.18)$$

Volume averaging (A.18) term-by-term, we have:

$$\overline{\frac{\partial}{\partial t}\rho_c u_\iota u_\jmath} = \frac{\partial}{\partial t}\overline{\rho_c u_\iota u_\jmath} + \frac{1}{V}\int_S \rho_c u_i u_j (\mathrm{v}_k n_k + \dot{r}) dS \qquad (A.19)$$

In the previous section, we showed $\overline{\rho_c u_\iota u_\jmath} = \overline{\rho_c}\tilde{u}_i\tilde{u}_j + \bar{\rho}_c\tilde{R}_{ij}$, so that (A.19) may be re-written as (A.20):

$$\overline{\frac{\partial}{\partial t}\rho_c u_\iota u_J} = \frac{\partial}{\partial t}\,\overline{\rho_c}\tilde{u}_i\tilde{u}_j + \frac{\partial}{\partial t}\bar{\rho}_c\tilde{R}_{ij} + \frac{1}{V}\int_S \rho_c u_i u_j(\mathrm{v}_k n_k + \dot{r})dS \qquad (A.20)$$

Averaging the second term in (A.18), we have:

$$\overline{\frac{\partial}{\partial x_k}\rho_c u_\iota u_j u_k} = \frac{\partial}{\partial x_k}\,\overline{\rho_c u_\iota u_j u_k} - \frac{1}{V}\int_S \rho_c u_i u_j u_k n_k dS \qquad (A.21)$$

The first term on the right-hand side of (A.21) can be decomposed further viz.

$$\overline{\rho_c u_\iota u_j u_k} = \overline{\rho_c(\tilde{u}_\iota + u_\iota'')\big(\tilde{u}_j + u_j''\big)(\tilde{u}_k + u_k'')} =$$

$$\overline{\rho_c}\tilde{u}_i\tilde{u}_j\tilde{u}_k + \overline{\rho_c u_\iota''\tilde{u}_j\tilde{u}_k} + \overline{\rho_c u_j''\tilde{u}_\iota\tilde{u}_k} + \overline{\rho_c u_k''\tilde{u}_\iota\tilde{u}_j} +$$

$$\overline{\rho_c\tilde{u}_\iota u_j''u_k''} + \overline{\rho_c\tilde{u}_j u_\iota''u_k''} + \overline{\rho_c\tilde{u}_k u_\iota''u_j''} + \overline{\rho_c u_\iota''u_j''u_k''}$$

$$= \overline{\rho_c}\tilde{u}_i\tilde{u}_j\tilde{u}_k + \tilde{u}_i\bar{\rho}_c\tilde{R}_{jk} + \tilde{u}_j\bar{\rho}_c\tilde{R}_{ik} + \tilde{u}_k\bar{\rho}_c\tilde{R}_{ij} + \overline{\rho_c u_\iota''u_j''u_k''} \qquad (A.22)$$

(A.21) can now be expressed as (A.23):

$$\overline{\frac{\partial}{\partial x_k}\rho_c u_\iota u_j u_k} = \frac{\partial}{\partial x_k}\overline{\rho_c}\tilde{u}_i\tilde{u}_j\tilde{u}_k + \frac{\partial}{\partial x_k}\big(\tilde{u}_i\bar{\rho}_c\tilde{R}_{jk} + \tilde{u}_j\bar{\rho}_c\tilde{R}_{ik} + \tilde{u}_k\bar{\rho}_c\tilde{R}_{ij}\big) + \frac{\partial}{\partial x_k}\,\overline{\rho_c u_\iota''u_j''u_k''} -$$

$$\frac{1}{V}\int_S \rho_c u_i u_j u_k n_k dS \qquad (A.23)$$

Continuing our application of volume averaging to (A.18), we now consider the pressure and deviatoric stress terms:

$$\overline{u_i \frac{\partial p}{\partial x_j}} = \overline{\frac{\partial}{\partial x_j} p u_i} - \overline{p \frac{\partial u_i}{\partial x_j}} = \frac{\partial}{\partial x_j} \overline{p u_i} - \frac{1}{V} \int_S p u_i n_j dS - \overline{p \frac{\partial u_i}{\partial x_j}} \quad (A.24)$$

$$\overline{u_j \frac{\partial p}{\partial x_i}} = \overline{\frac{\partial}{\partial x_i} p u_j} - \overline{p \frac{\partial u_j}{\partial x_i}} = \frac{\partial}{\partial x_i} \overline{p u_j} - \frac{1}{V} \int_S p u_j n_i dS - \overline{p \frac{\partial u_j}{\partial x_i}} \quad (A.25)$$

$$\overline{u_i \frac{\partial \tau_{jk}}{\partial x_k}} = \overline{\frac{\partial}{\partial x_k} \tau_{jk} u_i} - \overline{\tau_{jk} \frac{\partial u_i}{\partial x_k}} = \frac{\partial}{\partial x_k} \overline{\tau_{jk} u_i} - \frac{1}{V} \int_S \tau_{jk} u_i n_k dS - \overline{\tau_{jk} \frac{\partial u_i}{\partial x_k}} \quad (A.26)$$

$$\overline{u_j \frac{\partial \tau_{ik}}{\partial x_k}} = \overline{\frac{\partial}{\partial x_k} \tau_{ik} u_j} - \overline{\tau_{ik} \frac{\partial u_j}{\partial x_k}} = \frac{\partial}{\partial x_k} \overline{\tau_{ik} u_j} - \frac{1}{V} \int_S \tau_{ik} u_j n_k dS - \overline{\tau_{ik} \frac{\partial u_j}{\partial x_k}} \quad (A.27)$$

The gravity terms can be expressed as (A.28):

$$\overline{\rho_c u_i g_j} + \overline{\rho_c u_j g_i} = \bar{\rho}_c \left(g_i \tilde{u}_j + g_j \tilde{u}_i \right) \quad (A.28)$$

We now return our attention to the third and fourth terms on the right-hand side of (A.15):

$$\tilde{u}_i \cdot \overline{NS}_j = \tilde{u}_i \cdot \{ \frac{\partial}{\partial t} \alpha_c < \rho_c > \tilde{u}_j + \frac{\partial}{\partial x_k} \alpha_c < \rho_c > \tilde{u}_j \tilde{u}_k =$$

$$- \frac{\partial}{\partial k} \alpha_c < \rho_c > \tilde{R}_{jk} - \frac{\partial \bar{p}}{\partial x_j} + \frac{\partial \overline{\tau_{jk}}}{\partial x_k} + \alpha_c < \rho_c > g_j + L_j \} \quad (A.29)$$

Where $L_j = \frac{1}{V} \int_S \left(\rho_c u_j w + p n_j - \tau_{ik} n_k \right) dS$. Similarly, we have:

$$\tilde{u}_j \cdot \overline{NS}_i = \tilde{u}_j \cdot \{ \frac{\partial}{\partial t} \alpha_c < \rho_c > \tilde{u}_i + \frac{\partial}{\partial x_k} \alpha_c < \rho_c > \tilde{u}_i \tilde{u}_k =$$

$$- \frac{\partial}{\partial k} \alpha_c < \rho_c > \tilde{R}_{ik} - \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \overline{\tau_{ik}}}{\partial x_k} + \alpha_c < \rho_c > g_i + L_i \} \quad (A.30)$$

(A.29) and (A.30) may be combined using the product rule for differentiation to yield (A.31):

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > \tilde{u}_i\tilde{u}_j + \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{u}_i\tilde{u}_j\tilde{u}_k =$$

$$-\tilde{u}_i \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{R}_{jk} - \tilde{u}_j \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{R}_{ik}$$

$$-\tilde{u}_i \frac{\partial \bar{p}}{\partial x_j} - \tilde{u}_j \frac{\partial \bar{p}}{\partial x_i} + \tilde{u}_i \frac{\partial \overline{\tau_{jk}}}{\partial x_k} + \tilde{u}_j \frac{\partial \overline{\tau_{ik}}}{\partial x_k} + \bar{\rho}_c\left(g_i\tilde{u}_j + g_j\tilde{u}_i\right) + \tilde{u}_i L_j + \tilde{u}_j L_i \quad (A.31)$$

Combining (A.31) with (A.20), (A.23-28), (A.15) is re-written as (A.32):

$$\overline{u_i \cdot NS_j + u_j \cdot NS_i} - \left(\tilde{u}_i \cdot \overline{NS_j} + \tilde{u}_j \cdot \overline{NS_i}\right) \rightarrow$$

$$\frac{\partial}{\partial t}\alpha_c < \rho_c > \tilde{R}_{ij} + \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \left(\tilde{u}_i\tilde{R}_{jk} + \tilde{u}_j\tilde{R}_{ik} + \tilde{u}_k\tilde{R}_{ij}\right) + \frac{\partial}{\partial x_k}\overline{\rho_c u_i'' u_j'' u_k''} =$$

$$\tilde{u}_i \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{R}_{jk} + \tilde{u}_j \frac{\partial}{\partial x_k}\alpha_c < \rho_c > \tilde{R}_{ik}$$

$$-\frac{\partial}{\partial x_j}\overline{pu_i} + \overline{p\frac{\partial u_i}{\partial x_j}} + \tilde{u}_i \frac{\partial \bar{p}}{\partial x_j}$$

$$-\frac{\partial}{\partial x_i}\overline{pu_j} + \overline{p\frac{\partial u_j}{\partial x_i}} + \tilde{u}_j \frac{\partial \bar{p}}{\partial x_i}$$

$$+\frac{\partial}{\partial x_k}\overline{u_i\tau_{jk}} - \overline{\tau_{jk}\frac{\partial u_i}{\partial x_k}} - \tilde{u}_i \frac{\partial \overline{\tau_{jk}}}{\partial x_k}$$

$$+\frac{\partial}{\partial x_k}\overline{u_j\tau_{ik}} - \overline{\tau_{ik}\frac{\partial u_j}{\partial x_k}} - \tilde{u}_j \frac{\partial \overline{\tau_{ik}}}{\partial x_k}$$

$$S_{ij} - \tilde{u}_i L_j - \tilde{u}_j L_i \quad (A.32)$$

Here, S_{ij} is written as:

$$S_{ij} = \frac{1}{V}\int_S \left[\rho_c u_i u_j w + \left(p u_i n_j + p u_j n_i\right) - \left(u_i \tau_{jk} n_k + u_j \tau_{ik} n_k\right)\right] dS \qquad (A.33)$$

(A.32) is read as the transport equation for the density normalized generalized Reynolds stress tensor. While (A.32) is a valid form of the R-S equation, we will employ further algebra to express (A.32) in the more familiar form given in Schwarzkopf et al. [2]. Beginning with the triple-product term, we may write:

$$\frac{\partial}{\partial x_k}\,\overline{\rho_c u_i'' u_j'' u_k''} = \frac{\partial}{\partial x_k}\,\alpha_c < \rho_c u_i'' u_j'' u_k'' > \qquad (A.34)$$

Using $p = \bar{p} + p'$ and $u = \tilde{u} + u''$, the pressure terms may be decomposed viz.

$$\frac{\partial}{\partial x_j}\,\overline{p u_\iota} = \frac{\partial}{\partial x_j}\overline{\bar{p}\tilde{u}_\iota + p'\tilde{u}_\iota + \bar{p}u_\iota'' + p'u_\iota''} \qquad (A.35)$$

$$\overline{p\frac{\partial u_\iota}{\partial x_j}} = \overline{\bar{p}\frac{\partial}{\partial x_j}\tilde{u}_\iota + p'\frac{\partial}{\partial x_j}\tilde{u}_\iota + \bar{p}\frac{\partial}{\partial x_j}u_\iota'' + p'\frac{\partial}{\partial x_j}u_\iota''} \qquad (A.36)$$

$$\frac{\partial}{\partial x_i}\,\overline{p u_\jmath} = \frac{\partial}{\partial x_i}\overline{\bar{p}\tilde{u}_\jmath + p'\tilde{u}_\jmath + \bar{p}u_\jmath'' + p'u_\jmath''} \qquad (A.37)$$

$$\overline{p\frac{\partial u_\jmath}{\partial x_\iota}} = \overline{\bar{p}\frac{\partial}{\partial x_\iota}\tilde{u}_\jmath + p'\frac{\partial}{\partial x_\iota}\tilde{u}_\jmath + \bar{p}\frac{\partial}{\partial x_\iota}u_\jmath'' + p'\frac{\partial}{\partial x_\iota}u_\jmath''} \qquad (A.38)$$

An analogous decomposition is employed on the deviatoric terms yielding (A.36-39):

$$\frac{\partial}{\partial x_k}\,\overline{u_\iota \tau_{\jmath k}} = \frac{\partial}{\partial x_k}\overline{\tilde{\tau}_{\jmath k}\tilde{u}_\iota + \tau''_{\jmath k}\tilde{u}_\iota + \tilde{\tau}_{\jmath k}u''_\iota + \tau''_{\jmath k}u''_\iota} \tag{A.39}$$

$$\overline{\tau_{\jmath k}\frac{\partial u_\iota}{\partial x_k}} = \overline{\tilde{\tau}_{\jmath k}\frac{\partial}{\partial x_k}\tilde{u}_\iota + \tau''_{\jmath k}\frac{\partial}{\partial x_k}\tilde{u}_\iota + \tilde{\tau}_{\jmath k}\frac{\partial}{\partial x_k}u''_\iota + \tau''_{\jmath k}\frac{\partial}{\partial x_k}u''_\iota} \tag{A.40}$$

$$\frac{\partial}{\partial x_k}\,\overline{u_\jmath \tau_{\iota k}} = \frac{\partial}{\partial x_k}\overline{\tilde{\tau}_{\iota k}\tilde{u}_\jmath + \tau''_{\iota k}\tilde{u}_\jmath + \tilde{\tau}_{\iota k}u''_\jmath + \tau''_{\iota k}u''_\jmath} \tag{A.41}$$

$$\overline{\tau_{\iota k}\frac{\partial u_\jmath}{\partial x_k}} = \overline{\tilde{\tau}_{\iota k}\frac{\partial}{\partial x_k}\tilde{u}_\jmath + \tau''_{\iota k}\frac{\partial}{\partial x_k}\tilde{u}_\jmath + \tilde{\tau}_{\iota k}\frac{\partial}{\partial x_k}u''_\jmath + \tau''_{\iota k}\frac{\partial}{\partial x_k}u''_\jmath} \tag{A.42}$$

The next part of the derivation frequently employs the relationship (9) noting that combining like terms means one but not both of the following strategies may be employed: all derivatives appear under the average or derivatives are taken after employing the average. For this reason, it is not yet possible to combine terms in (A.35) and (A.36) for example. In the absence of dispersed phase effects we recall that the volume average and the derivative will commute. If (A.35) and (A.36) are to be reducible, we must allow for the derivative and the averaging to commute, but also include the integral effect due to the dispersed phase using an analog of (9). With this note of caution, we begin reducing (A.35-42) paying careful attention to signs in (A.32):

$$(A.36) - (A.35)\text{:} \quad -\overline{\left(\tilde{u}_\iota\frac{\partial \bar{p}}{\partial x_\jmath} + \tilde{u}_\iota\frac{\partial p'}{\partial x_\jmath} + u''_\iota\frac{\partial \bar{p}}{\partial x_\jmath} + u''_\iota\frac{\partial p'}{\partial x_\jmath}\right)} - B_{ij} \tag{A.43}$$

Where $B_{ij} = \frac{1}{V}\int_S [\bar{p}\tilde{u}_i + p'\tilde{u}_i + \bar{p}u''_i + p'u''_i]n_j\,dS = \frac{1}{V}\int_S pu_i n_j\,dS$ is the residual from switching the order of differentiation and averaging in (A.35). (A.37) and (A.38) may be combined in a similar fashion to yield (A.44):

$$(A.38) - (A.37)\text{:} \quad -\overline{\left(\tilde{u}_\jmath\frac{\partial \bar{p}}{\partial x_\iota} + \tilde{u}_\jmath\frac{\partial p'}{\partial x_\iota} + u''_\jmath\frac{\partial \bar{p}}{\partial x_\iota} + u''_\jmath\frac{\partial p'}{\partial x_\iota}\right)} - B_{ji} \tag{A.44}$$

Combing (A.43) and (A.44) with the remaining respective like-terms in (A.32), we have:

$$\tilde{u}_i \left[\overline{\frac{\partial p}{\partial x_j}} + \frac{1}{V}\int_S pn_j \, dS\right] - \overline{u_\iota \frac{\partial p}{\partial x_j}} - B_{ij} \qquad (A.45)$$

$$\tilde{u}_j \left[\overline{\frac{\partial p}{\partial x_\iota}} + \frac{1}{V}\int_S pn_i \, dS\right] - \overline{u_j \frac{\partial p}{\partial x_\iota}} - B_{ji} \qquad (A.46)$$

(A.45) and (A.46) may be written respectfully as (A.47) and (A.48):

$$-\overline{u_\iota'' \frac{\partial p}{\partial x_j}} + \tilde{u}_i \frac{1}{V}\int_S pn_j \, dS - B_{ij} = -\overline{u_\iota'' \frac{\partial p}{\partial x_j}} + Z_{ij} \qquad (A.47)$$

$$-\overline{u_j'' \frac{\partial p}{\partial x_\iota}} + Z_{ji} \qquad (A.48)$$

(A.47) and (A.48) may be written respectively as (A.49) and (A.50):

$$-\overline{u_\iota'' \frac{\partial p}{\partial x_j}} + Z_{ij} = \overline{p' \frac{\partial u_\iota''}{\partial x_j}} - \overline{\frac{\partial \bar{p}}{\partial x_j} u_\iota''} - \overline{\frac{\partial}{\partial x_j} p' u_\iota''} + Z_{ij} \qquad (A.49)$$

$$-\overline{u_j'' \frac{\partial p}{\partial x_\iota}} + Z_{ji} = \overline{p' \frac{\partial u_j''}{\partial x_\iota}} - \overline{\frac{\partial \bar{p}}{\partial x_\iota} u_j''} - \overline{\frac{\partial}{\partial x_\iota} p' u_j''} + Z_{ji} \qquad (A.50)$$

Defining $a_i = -\overline{u_\iota''}$, and using the property that $\overline{\frac{\partial \bar{A}}{\partial x_\iota} B''} = \frac{\partial \bar{A}}{\partial x_i} \overline{B''}$ [8], the final expressions for the pressure terms can be written as (A.51):

[8] This property is not obvious and a good exercise for the interested reader. Hint: use product rule and apply derivative-average commuting twice. The integral expressions will cancel.

$$\overline{p'\frac{\partial u_i''}{\partial x_j}} + \overline{p'\frac{\partial u_j''}{\partial x_i}} + a_i\frac{\partial \bar{p}}{\partial x_j} + a_j\frac{\partial \bar{p}}{\partial x_i} - \frac{\partial}{\partial x_j}\overline{p'u_i''} - \frac{\partial}{\partial x_i}\overline{p'u_j''}$$

$$+ Z_{ij} + Z_{ji} + \frac{1}{V}\int_S p'\left(u_j''n_i + u_i''n_j\right)dS \qquad (A.51)$$

The pressure terms are now in the form given in Schwarzkopf et al. [2]. We now perform analogous algebra on the deviatoric terms as that used on the pressure terms. Combining (A.39) with (A.40) and (A.41) with (A.42), we have:

$$\overline{\tilde{u}_i\frac{\partial \tilde{\tau}_{jk}}{\partial x_k} + \tilde{u}_i\frac{\partial \tau_{jk}''}{\partial x_k} + u_i''\frac{\partial \tilde{\tau}_{jk}}{\partial x_k} + u_i''\frac{\partial \tau_{jk}''}{\partial x_k}} + \frac{1}{V}\int_S u_i\tau_{jk}n_k\,dS \qquad (A.52)$$

$$\overline{\tilde{u}_j\frac{\partial \tilde{\tau}_{ik}}{\partial x_k} + \tilde{u}_j\frac{\partial \tau_{ik}''}{\partial x_k} + u_j''\frac{\partial \tilde{\tau}_{ik}}{\partial x_k} + u_j''\frac{\partial \tau_{ik}''}{\partial x_k}} + \frac{1}{V}\int_S u_j\tau_{ik}n_k\,dS \qquad (A.53)$$

(A.52) and (A.53) may be combined with their respective remaining counterparts in (A.32) to yield:

$$\overline{u_i''\frac{\partial \tau_{jk}}{\partial x_k}} + \frac{1}{V}\int_S u_i''\tau_{jk}n_k\,dS \qquad (A.54)$$

$$\overline{u_j''\frac{\partial \tau_{ik}}{\partial x_k}} + \frac{1}{V}\int_S u_j''\tau_{ik}n_k\,dS \qquad (A.55)$$

More algebra yields the deviatoric terms expressed in the form given in [2]:

$$\frac{\partial}{\partial x_k}\,\overline{u_i''\tau_{jk}''} - a_i\frac{\partial\tilde{\tau}_{jk}}{\partial x_k} - \overline{\tau_{jk}''\frac{\partial u_i''}{\partial x_k}} + \frac{1}{V}\int_S u_i''\tilde{\tau}_{jk}n_k\,dS \qquad (A.56)$$

$$\frac{\partial}{\partial x_k}\,\overline{u_j''\tau_{ik}''} - a_j\frac{\partial\tilde{\tau}_{ik}}{\partial x_k} - \overline{\tau_{ik}''\frac{\partial u_j''}{\partial x_k}} + \frac{1}{V}\int_S u_j''\tilde{\tau}_{ik}n_k\,dS \qquad (A.57)$$

Combining new integral terms with those in (A.32), the Reynolds stress equation expressed in a form analogous to its presentation in [2] may now be written:

$$\begin{aligned}
&\frac{\partial}{\partial t}\alpha_c<\rho_c>\tilde{R}_{ij} + \frac{\partial}{\partial x_k}\alpha_c<\rho_c>\tilde{u}_k\tilde{R}_{ij} + \frac{\partial}{\partial x_k}\,\alpha_c<\rho_c u_i''u_j''u_k''>= \\
&\qquad -\alpha_c<\rho_c>\tilde{R}_{jk}\frac{\partial\tilde{u}_i}{\partial x_k} - \alpha_c<\rho_c>\tilde{R}_{ik}\frac{\partial\tilde{u}_j}{\partial x_k} \\
&\qquad -\frac{\partial}{\partial x_j}\alpha_c<p'u_i''> + \alpha_c<p'\frac{\partial u_i''}{\partial x_j}> + a_i\frac{\partial\bar{p}}{\partial x_j} \\
&\qquad -\frac{\partial}{\partial x_i}\,\alpha_c<p'u_j''> + \alpha_c<p'\frac{\partial u_j''}{\partial x_i}> + a_j\frac{\partial\bar{p}}{\partial x_i} \\
&\qquad +\frac{\partial}{\partial x_k}\,\alpha_c<u_i''\tau_{jk}''> - \alpha_c<\tau_{jk}''\frac{\partial u_i''}{\partial x_k}> - a_i\frac{\partial\tilde{\tau}_{jk}}{\partial x_k} \\
&\qquad +\frac{\partial}{\partial x_k}\,\alpha_c<u_j''\tau_{ik}''> - \alpha_c<\tau_{ik}''\frac{\partial u_j''}{\partial x_k}> - a_j\frac{\partial\tilde{\tau}_{ik}}{\partial x_k} \\
&\qquad + E_{ij} + P_{ij} - T_{ij}
\end{aligned} \qquad (A.58)$$

Where the additional source terms E_{ij}, P_{ij}, T_{ij}, are dispersed phase effects owing to mass-transfer, pressure, and viscous contributions, respectfully, and in the most reduced form are expressed:

$$\boxed{E_{ij} = \frac{1}{V}\int_S \left[\rho_c w\left(u_i'' u_j'' - \tilde{u}_i \tilde{u}_j\right)\right] dS} \qquad (A.59)$$

$$\boxed{P_{ij} = \frac{1}{V}\int_S \left(p' u_i'' n_j + p' u_j'' n_i\right) dS} \qquad (A.60)$$

$$\boxed{T_{ij} = \frac{1}{V}\int_S \left(u_i'' \tau_{jk}'' + u_j'' \tau_{ik}''\right) n_k \, dS} \qquad (A.61)$$

Derivation of Turbulent Mass Flux Equation

The turbulent mass flux equation is derived by first developing the transport equation for the mass-averaged fluctuating velocity $u''$and then applying volume averaging to the resulting equation. The u''equation is obtained by subtracting the volume-averaged Navier-Stokes equation (A.14) from the instantaneous Navier-Stokes equation (A.6) viz.

$$\left[\frac{\partial}{\partial t}\rho_c u_i + \frac{\partial}{\partial x_j}\rho_c u_i u_j = \frac{\partial}{\partial x_j}\sigma_{ij} + \rho_c g_i\right] -$$

$$\left[\frac{\partial}{\partial t}\overline{\rho_c}\tilde{u}_i + \frac{\partial}{\partial x_j}\overline{\rho_c}\tilde{u}_i\tilde{u}_j = -\frac{\partial}{\partial x_j}\overline{\rho_c}\tilde{R}_{ij} + \frac{\partial\overline{\sigma_{ij}}}{\partial x_i} + \overline{\rho_c}g_i + C_i\right] \quad (A.62)$$

Where C_i is the integral expression defined in (A.14). Using the procedure discussed in Besnard et al. [1], we use continuity and group like-terms to obtain the u'' equation:

$$\overline{\rho_c}\frac{\partial u_i''}{\partial t} + \overline{\rho_c}\tilde{u}_j\frac{\partial u_i''}{\partial x_j} + \overline{\rho_c}u_j''\frac{\partial u_i}{\partial x_j} = \frac{\partial}{\partial x_j}\overline{\rho_c}\tilde{R}_{ij} + \frac{\overline{\rho_c}}{\rho_c}\frac{\partial\sigma_{ij}}{\partial x_j} - \frac{\partial\overline{\sigma_{ij}}}{\partial x_i} + (\tilde{u}_i\cdot\chi - C_i) \quad (A.63)$$

Here, χ is the continuity source term defined in (A.5). Applying volume averaging to (A.63) term-by-term, we have:

$$\overline{\overline{\rho_c}\frac{\partial u_i''}{\partial t}} = \overline{\rho_c}\left[\frac{\partial\overline{u_i''}}{\partial t} + \frac{1}{V}\int_S u_i''(\mathrm{v}_k n_k + \dot{r})dS\right] \quad (A.64)$$

$$\overline{\overline{\rho_c}\tilde{u}_j\frac{\partial u_i''}{\partial x_j}} = \overline{\rho_c}\tilde{u}_j\left[\frac{\partial\overline{u_i''}}{\partial x_j} - \frac{1}{V}\int_S u_i'' n_j dS\right] \quad (A.65)$$

$$\overline{\overline{\rho_c}u_j''\frac{\partial u_i}{\partial x_j}} = \overline{\rho_c}\left(\overline{u_j''\frac{\partial\bar{u}_i}{\partial x_j}} + \overline{u_j''\frac{\partial u_i'}{\partial x_j}}\right) \quad = \quad \overline{\rho_c}\left(\overline{u_j''}\frac{\partial\bar{u}_i}{\partial x_j} + \overline{u_j''\frac{\partial u_i'}{\partial x_j}}\right) \quad (A.66)$$

Using identity (10) in [1], the last term in (A.66) may be decomposed in terms of the Reynolds stress and the turbulent mass flux so that the whole of (A.66) becomes:

$$\overline{\overline{\rho_c} u_j'' \frac{\partial u_\iota}{\partial x_j}} =$$

$$\overline{\rho_c} \cdot \left(\overline{u_j''} \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial}{\partial x_j} \left(a_i a_j + \tilde{R}_{ij} - \overline{\rho' u_\iota' u_j'} / \overline{\rho_c} \right) - \frac{1}{V} \int_S u_\iota' u_j'' n_j dS - \overline{u_\iota' \frac{\partial u_j''}{\partial x_j}} \right) \quad (A.67)$$

Returning to (A.63), the average of the stress terms may be expressed as (A.68):

$$\overline{\frac{\overline{\rho_c}}{\rho_c} \frac{\partial \sigma_{\iota J}}{\partial x_J}} - \overline{\frac{\partial \overline{\sigma_{\iota J}}}{\partial x_\iota}} = \overline{\left(\frac{\overline{\rho_c}}{\rho_c} - 1 \right) \frac{\partial \overline{\sigma_{\iota J}}}{\partial x_\iota}} + \overline{\frac{\overline{\rho_c}}{\rho_c} \frac{\partial \sigma_{\iota J}'}{\partial x_\iota}} = b \frac{\partial \overline{\sigma_{\iota J}}}{\partial x_i} + \overline{\rho_c} \left\{ \overline{\left(\frac{1}{\rho_c} \right)} \overline{\frac{\partial \sigma_{\iota J}'}{\partial x_\iota}} + \overline{\left(\frac{1}{\rho_c} \right)' \frac{\partial \sigma_{\iota J}'}{\partial x_\iota}} \right\} =$$

$$b \frac{\partial \overline{\sigma_{\iota J}}}{\partial x_i} + \overline{\rho_c} \overline{\left(\frac{1}{\rho_c} \right)' \frac{\partial \sigma_{\iota J}'}{\partial x_\iota}} - (b+1) \frac{1}{V} \int_S \sigma_{ij}' n_j dS \quad (A.68)$$

The remaining terms can be expressed as (A.69)[9] and (A.70)[10]:

$$\overline{\frac{\partial}{\partial x_J} \overline{\rho_c} \tilde{R}_{\iota J}} = \frac{\partial}{\partial x_j} \overline{\rho_c} \tilde{R}_{ij} \quad (A.69)$$

And,

$$\overline{(\tilde{u}_\iota \cdot \chi - C_\iota)} = not\ reducible \quad (A.70)$$

Collecting terms in (A.64)-(A.70), we have (A.71):

$$\overline{\rho_c} \frac{\partial \overline{u_\iota''}}{\partial t} + \overline{\rho_c} \tilde{u}_j \frac{\partial \overline{u_\iota''}}{\partial x_j} + \overline{\rho_c} \overline{u_j''} \frac{\partial \bar{u}_i}{\partial x_j} + \overline{\rho_c} \frac{\partial}{\partial x_j} \left(a_i a_j + \tilde{R}_{ij} - \frac{\overline{\rho' u_\iota' u_j'}}{\overline{\rho_c}} \right) - \overline{\rho_c} \overline{u_\iota' \frac{\partial u_j''}{\partial x_j}} =$$

$$b \frac{\partial \overline{\sigma_{\iota J}}}{\partial x_i} + \overline{\rho_c} \overline{\left(\frac{1}{\rho_c} \right)' \frac{\partial \sigma_{\iota J}'}{\partial x_\iota}} + \frac{\partial}{\partial x_j} \overline{\rho_c} \tilde{R}_{ij} + B_i \quad (A.71)$$

[9] The integral term in (A.69) is zero because the mean terms in the integrand will be constant along the path. In other words, mean properties are invariant everywhere within the control volume so they are also invariant along particle surfaces within the volume.

[10] Proving (A.70) to be zero or non-zero is an open problem.

Where B_i is the sum of the integral contributions due to the dispersed phase (A.72):

$$B_i = \overline{(\tilde{u}_\iota \cdot \chi - C_\iota)} + \overline{\rho_c}\tilde{u}_j \frac{1}{V}\int_S u_i'' n_j dS - \overline{\rho_c}\frac{1}{V}\int_S u_i''(\mathrm{v}_j n_j + \dot{r})dS$$

$$+\overline{\rho_c}\frac{1}{V}\int_S u_i' u_j'' n_j dS - (b+1)\frac{1}{V}\int_S \sigma_{ij}' n_j dS \quad (A.72)$$

Observe that B_i may be re-written in a simpler form:

$$\boxed{B_i = \overline{(\tilde{u}_\iota \cdot \chi - C_\iota)} + \overline{\rho_c}\frac{1}{V}\int_S u_i'' w n_j n_j dS - (b+1)\frac{1}{V}\int_S \sigma_{ij}' n_j dS} \quad (A.73)$$

Defining the turbulent mass flux $a_i = -\overline{u_\iota''}$ and using continuity, (A.71) is cast into a form similar to that which is presented in [2] (A.74):

$$\frac{\partial \overline{\rho_c} a_i}{\partial t} + \frac{\partial \overline{\rho_c}\tilde{u}_j a_i}{\partial x_j}$$
$$= -b\frac{\partial \overline{\sigma_{\iota J}}}{\partial x_i} - \tilde{R}_{ij}\frac{\partial \overline{\rho_c}}{\partial x_j} + \overline{\rho_c}\frac{\partial}{\partial x_j}a_i a_j - \overline{\rho_c} a_j \frac{\partial \overline{u}_i}{\partial x_j} - \overline{\rho_c}\frac{\partial}{\partial x_j}\left(\frac{\overline{\rho' u_i' u_j'}}{\overline{\rho_c}}\right) - \overline{\rho_c}\overline{\left(\frac{1}{\rho_c}\right)' \frac{\partial \sigma_{\iota J}'}{\partial x_\iota}}$$
$$-\overline{\rho_c}\overline{u_\iota' \frac{\partial u_j''}{\partial x_j}} - B_i + a_i \chi \quad (A.74)$$

Note the final non-integral term on the right-hand side of (A.74) is a dilatation term that was neglected in the original BHR derivation [1]. This term would still exist for a single-phase compressible flow, but its magnitude may be small depending on the application.

Derivation of Density-Specific Volume Covariance Equation

We derive the density-specific volume covariance equation. Beginning with the instantaneous continuity equation (A.1), we re-write $\rho_c = 1/\nu$, yielding the conservation of mass relationship in specific-volume form (A.75):

$$\frac{\partial \nu}{\partial t} + u_i \frac{\partial \nu}{\partial x_i} = \nu \frac{\partial u_i}{\partial x_i} \qquad (A.75)$$

where ν is the instantaneous specific volume. Multiplying (A.75) by instantaneous density and re-writing in conservative form, we have (A.76):

$$\frac{\partial \rho_c \nu}{\partial t} + \frac{\partial \rho_c \nu u_i}{\partial x_i} = \rho_c \nu \frac{\partial u_i}{\partial x_i} \qquad (A.76)$$

It is worth noting that $\rho_c \nu = 1$ by definition. For the moment, we will not apply the instantaneous density-specific volume identity and (A.76) will be combined with a later equation to yield the density-specific volume covariance equation. This strategy will simplify the resulting algebra in the derivation. An alternate strategy to arrive at the b-equation was proposed in [1]. We will show the two approaches yield the same result in the limit dispersed phase effects vanish. Continuing the derivation, we multiply (A.75) by the volume-averaged density $\overline{\rho_c}$, re-writing the result in conservative form we have (A.77):

$$\frac{\partial \overline{\rho_c} \nu}{\partial t} + \frac{\partial \overline{\rho_c} \nu u_i}{\partial x_i} - \nu\chi - \nu \frac{\partial \overline{\rho_c} u_i''}{\partial x_i} = \overline{\rho_c} \nu \frac{\partial u_i}{\partial x_i} \qquad (A.77)$$

Where the third and fourth term on the left-hand side of (A.77) come from the filtered continuity equation (A.5) and the Favre velocity decomposition. The b-equation can now be obtained by volume averaging term-by-term in (A.76) and (A.77) respectively, and then taking the difference of the resulting averaged equations. Beginning with (A.76), the filtered terms are (A.78-80):

$$\overline{\frac{\partial \rho_c \nu}{\partial t}} = \frac{\partial}{\partial t}\left(\overline{\rho_c}\bar{\nu} + \overline{\rho'\nu'}\right) + \frac{1}{V}\int_S \rho_c \nu(\mathrm{v}_i n_i + \dot{r})dS \quad (A.78)$$

$$\overline{\frac{\partial \rho_c \nu u_\iota}{\partial x_\iota}} = \frac{\partial}{\partial x_i}\left(\overline{\rho_c}\bar{\nu}\bar{u}_i + \bar{u}_i\overline{\rho'\nu'} + \overline{\rho_c}\overline{\nu' u_\iota{}'} + \bar{\nu}\overline{\rho' u_\iota{}'} + \overline{\rho'\nu' u_\iota{}'}\right) - \frac{1}{V}\int_S \rho_c \nu u_i n_i dS \quad (A.79)$$

$$\overline{\rho_c \nu \frac{\partial u_\iota}{\partial x_\iota}} = \overline{\rho_c}\bar{\nu}\frac{\partial \bar{u}_i}{\partial x_i} + \overline{\rho_c}\overline{\nu'\frac{\partial u_\iota'}{\partial x_\iota}} + \bar{\nu}\overline{\rho'\frac{\partial u_\iota'}{\partial x_\iota}} + \overline{\rho'\nu'\frac{\partial \bar{u}_\iota}{\partial x_\iota}} + \overline{\rho'\nu'\frac{\partial u_\iota'}{\partial x_\iota}} - \frac{1}{V}\int_S \overline{\rho_c}\bar{\nu}u_i n_i dS \quad (A.80)$$

We now apply volume-averaging term-by-term to (A.77) yielding (A.81-85)[11]:

$$\overline{\frac{\partial \overline{\rho_c}\nu}{\partial t}} = \frac{\partial}{\partial t}\overline{\rho_c}\bar{\nu} + \frac{1}{V}\int_S \overline{\rho_c}\nu(\mathrm{v}_i n_i + \dot{r})dS \quad (A.81)$$

$$\overline{\frac{\partial \overline{\rho_c}\nu u_\iota}{\partial x_\iota}} = \frac{\partial}{\partial x_i}\left(\overline{\rho_c}\bar{\nu}\bar{u}_i + \overline{\rho_c}\overline{\nu' u_\iota{}'}\right) - \frac{1}{V}\int_S \overline{\rho_c}\nu u_i n_i dS \quad (A.82)$$

$$-\overline{\nu}\overline{\chi} = not\ reducible \quad (A.83)$$

$$-\overline{\nu\frac{\partial \overline{\rho_c}u_\iota''}{\partial x_\iota}} = \bar{\nu}\frac{\partial}{\partial x_i}\overline{\rho_c}a_i - \overline{\nu'\frac{\partial}{\partial x_\iota}\overline{\rho_c}u_\iota''} + \overline{\rho_c}\bar{\nu}\frac{1}{V}\int_S u_i'' n_i dS \quad (A.84)$$

$$\overline{\overline{\rho_c}\nu\frac{\partial u_\iota}{\partial x_\iota}} = \overline{\rho_c}\bar{\nu}\frac{\partial \bar{u}_i}{\partial x_i} + \overline{\rho_c}\overline{\nu'\frac{\partial u_\iota'}{\partial x_\iota}} - \frac{1}{V}\int_S \overline{\rho_c}\bar{\nu}u_i n_i dS \quad (A.85)$$

Subtracting (A.78-80) from (A.81-85), and defining the density-specific volume correlation $b = \bar{\rho}_c\bar{\nu} - 1 = -\overline{\rho'\nu'}$, we have:

$$\frac{\partial b}{\partial t} + \frac{\partial \bar{u}_i b}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\bar{\nu}\overline{\rho' u_\iota{}'} + \overline{\rho'\nu' u_\iota{}'}\right) + \bar{\nu}\frac{\partial}{\partial x_i}\overline{\rho_c}a_i - \overline{\nu'\frac{\partial}{\partial x_\iota}\overline{\rho_c}u_\iota''} + \bar{\nu}\overline{\rho'\frac{\partial u_\iota'}{\partial x_\iota}} + \overline{\rho'\nu'\frac{\partial \bar{u}_\iota}{\partial x_\iota}} + \overline{\rho'\nu'\frac{\partial u_\iota'}{\partial x_\iota}} = Q \quad (A.86)$$

[11] Proving (A.83) zero or non-zero is an open problem.

Where,

$$Q = \frac{1}{V}\int_S \rho_c' \nu(\mathrm{v}_i n_i + \dot{r}) dS - \frac{1}{V}\int_S \rho_c' \nu u_i n_i dS + \overline{\nu\chi} - \overline{\rho_c}\bar{\nu}\frac{1}{V}\int_S u_i'' n_i dS \qquad (A.87)$$

Some algebra is needed to write (A.86) in the form expressed in [2]. We may note (A.88) and (A.89):

$$\frac{\partial \bar{u}_i b}{\partial x_i} + \overline{\rho'\nu'\frac{\partial \bar{u}_\iota}{\partial x_\iota}} = \frac{\partial \bar{u}_i b}{\partial x_i} + \overline{\rho'\nu'}\frac{\partial \bar{u}_i}{\partial x_i} = \frac{\partial \bar{u}_i b}{\partial x_i} - b\frac{\partial \bar{u}_i}{\partial x_i} = \bar{u}_i \frac{\partial b}{\partial x_i} \qquad (A.88)$$

$$\bar{\nu}\ \frac{\partial}{\partial x_i}\overline{\rho_c} a_i = \left(\frac{b+1}{\bar{\rho}_c}\right)\frac{\partial}{\partial x_i}\overline{\rho_c} a_i \qquad (A.89)$$

(A.86) becomes (A.90):

$$\frac{\partial b}{\partial t} + \bar{u}_i \frac{\partial b}{\partial x_i} + \left(\frac{b+1}{\bar{\rho}_c}\right)\frac{\partial}{\partial x_i}\overline{\rho_c} a_i - Q =$$

$$\frac{\partial}{\partial x_i}\left(\bar{\nu}\overline{\rho' u_\iota{}'} + \overline{\rho'\nu' u_\iota{}'}\right) + \overline{\nu'\frac{\partial}{\partial x_\iota}\overline{\rho_c} u_\iota''} - \bar{\nu}\overline{\rho'\frac{\partial u_\iota'}{\partial x_\iota}} - \overline{\rho'\nu'\frac{\partial u_\iota'}{\partial x_\iota}} \qquad (A.90)$$

The right-hand side of (A.90) requires more work. After considerable algebra, (A.90) may be expressed in the form presented in (A.91) [2]:

$$\boxed{\frac{\partial b}{\partial t} + \bar{u}_i \frac{\partial b}{\partial x_i} + \left(\frac{b+1}{\bar{\rho}_c}\right)\frac{\partial}{\partial x_i}\overline{\rho_c} a_i + \overline{\rho_c}\frac{\partial}{\partial x_i}\overline{\nu' u_\iota{}'} - 2\overline{\rho_c}\overline{\nu'\frac{\partial u_\iota'}{\partial x_\iota}} = H} \qquad (A.91)$$

Where H is the net source of b owing to dispersed phase effects and is written:

$$H = Q - b\frac{1}{V}\int_S u_i' n_i dS$$

$$= \frac{1}{V}\int_S \rho_c' v(\mathrm{v}_i n_i + \dot{r})dS - \frac{1}{V}\int_S \rho_c' v u_i n_i dS + \overline{v\chi} - \overline{\rho_c}\overline{v}\frac{1}{V}\int_S u_i'' n_i dS - b\frac{1}{V}\int_S u_i' n_i dS$$

$$= -\frac{1}{V}\int_S \rho_c' v w n_i n_i dS + \overline{v\chi} - \overline{\rho_c}\overline{v}\frac{1}{V}\int_S u_i'' n_i dS - b\frac{1}{V}\int_S u_i' n_i dS$$

$$= -\frac{1}{V}\int_S \rho_c' v w n_i n_i dS + \overline{v\chi} - (b+1)\frac{1}{V}\int_S (u_i' - a_i) n_i dS - b\frac{1}{V}\int_S u_i' n_i dS$$

$$\boxed{H = -\frac{1}{V}\int_S \rho_c' v w n_i n_i dS + \overline{v\chi} - (2b+1)\frac{1}{V}\int_S u_i' n_i dS} \qquad (A.92)$$

LA-UR-13-26502
Approved for public release; distribution is unlimited.

Title: Modeling Kelvin-Helmholtz and Rayleigh-Taylor driven Mixing Layers using the BHR model

Author(s): Tan-Torres, Sasha A.

Intended for: Computational Physics Summer Workshop, 2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)
Report

Issued: 2013-08-16

Modeling Kelvin-Helmholtz and Rayleigh-Taylor driven Mixing Layers using the BHR model

Sasha Tan-Torres

August 15, 2013

Abstract

Turbulence models are often applied to problems that are that are self-similar. Calibrating and validating turbulence models outside of the self-similar regime of a mixing layer is important to consider, since models can sometimes offer a local fit, i.e. only the self-similar regime, rather than a global fit, inclusive of the transient regime. Combined buoyancy and shear driven mixing layers (due to Rayleigh-Taylor and Kelvin-Helmholtz instabilities respectively) are difficult to validate due to their inherent transient behavior, as well as their sensitivity to initial conditions. Using the BHR-3 model, we can attempt to model the behavior of the combined mixing layer and then compare the results to experimental data provided by a group at Texas A & M University.

1 Introduction

Coupled buoyancy and shear driven mixing occurs in multiple environments, both natural and manufactured. Some examples include oceans and rivers, mixing chambers, and inertial confinement fusion (ICF). When fluids of different species or density overlay one another, the interface is unstable and causes the two fluids to mix, which drives Rayleigh-Taylor instability. The buoyancy creates a normal force at the interface. In contrast, parallel streams of unequal velocity fluids create a tangential shear at the interface. This is known as Kelvin-Helmholtz instability. [1] When studying turbulent models, it is better to start with simple flows with easily targeted physics. Problems showing self-similar behavior at late time make for easy comparison to experimental data, regardless of the initial conditions. Unfortunately, this can often lead to models which represent the local behavior of the flow (generally in the self-similar region), but sacrifice global accuracy, such as the transient portion of the flow. The general mixing layer growth driven by both shear and buoyancy is a transient process. Though a model can be calibrated for shear or buoyancy separately, validation is difficult for when they are combined, since the transient effect of initial conditions and flow physics are hard to disentangle. The flow is very sensitive to initial conditions which makes calibration in the transient regime a challenge. The BHR-3 model is used to compare against experimental data and the results are discussed in this paper.

2 BHR-3 Model

BHR-3 is a model for shear and buoyancy driven mixing. It is a set of equations developed at Los Alamos National Laboratory. It is derived from the Navier Stokes equations and models variable density flow. [4] The equations are shown in (1–4), where ρ is density, $\tilde{R}_{ij}$ is Reynolds stress, $\tilde{u}$ is velocity, P is pressure, a_i is turbulent mass flux, S is the turbulent length scale, C_{r1-4}, C_{a1}, C_a, C_b, C_{b1}, C_s, and C_{1-4} are turbulent coefficients, and K is the turbulent kinetic energy.

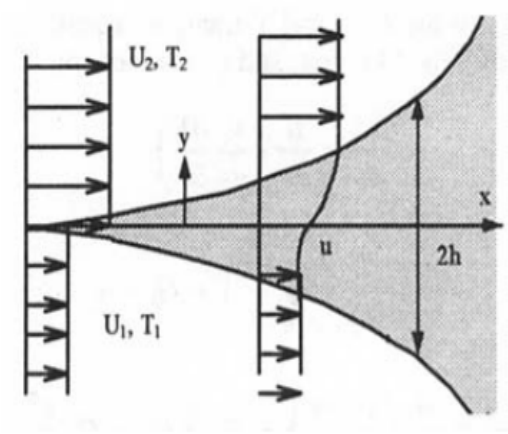


Figure 1: Buoyancy and Shear Mixing layer (taken from [3])

$$\begin{aligned}\frac{\partial\left(\bar{\rho}\tilde{R}_{ij}\right)}{\partial t}+\frac{\partial}{\partial x_k}\left(\bar{\rho}\check{u_k}\tilde{R}_{ij}\right) = &\ (1-C_{r1})\left[a_i\frac{\partial\bar{P}}{\partial x_j}+a_j\frac{\partial\bar{P}}{\partial x_i}\right]+\bar{\rho}(C_{r2}-1)\left[\tilde{R}_{ik}\frac{\partial\tilde{u}_j}{\partial x_k}+\tilde{R}_{jk}\frac{\partial\tilde{u}_i}{\partial x_k}\right]\\ &+C_{r3}\frac{\partial}{\partial x_k}\left(\frac{S}{\sqrt{K}}\bar{\rho}\tilde{R}_{km}\frac{\partial\tilde{R}_{ij}}{\partial x_m}\right)-C_{r4}\bar{\rho}\frac{\sqrt{K}}{S}\left(\tilde{R}_{ij}-\frac{1}{3}\tilde{R}_{kk}\delta_{ij}\right)\quad,\\ &-C_{r2}\frac{2}{3}\bar{\rho}\tilde{R}_{mk}\frac{\partial\tilde{u}_m}{\partial x_k}\delta_{ij}+C_{r1}\frac{2}{3}a_k\frac{\partial\bar{P}}{\partial x_k}\delta_{ij}-\bar{\rho}\frac{2}{3}\frac{K\sqrt{K}}{S}\delta_{ij}\end{aligned}\tag{1}$$

$$\begin{aligned}\frac{\partial\left(\bar{\rho}a_i\right)}{\partial t}+\frac{\partial}{\partial x_k}\left(\bar{\rho}\tilde{u}_k a_i\right) = &\ b\frac{\partial\bar{P}}{\partial x_i}-\tilde{R_{ik}}\frac{\partial\bar{\rho}}{\partial x_k}-\bar{\rho}a_k\frac{\partial\left(\tilde{u}_i-a_i\right)}{\partial x_k}+\bar{\rho}\frac{\partial}{\partial x_k}(a_k a_i)\\ &+C_a\bar{\rho}\frac{\partial}{\partial x_m}\left(\frac{S}{\sqrt{K}}R_{mn}\frac{\partial a_i}{\partial x_n}\right)-C_{a1}\bar{\rho}\frac{\sqrt{K}}{S}a_i\end{aligned}\tag{2}$$

$$\frac{\partial(\bar{\rho}b)}{\partial t}+\frac{\partial(\bar{\rho}b\tilde{u}_k}{\partial x_k}=2\bar{\rho}a_k\frac{\partial b}{\partial x_k}-2(b+1)a_k\frac{\partial\bar{\rho}}{\partial x_k}+C_b\bar{\rho}^2\frac{\partial}{\partial x_m}\left(\frac{S}{\bar{\rho}\sqrt{K}}R_{mn}\frac{\partial b}{\partial x_n}\right)-C_{b2}\bar{\rho}\frac{\sqrt{K}}{S}b\tag{3}$$

$$\begin{aligned}\frac{\partial\bar{\rho}S}{\partial t}+\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j S) = &\ \frac{S}{K}\left(\frac{3}{2}-C_4\right)a_j\frac{\partial\bar{P}}{\partial x_j}-\frac{S}{K}\left(\frac{3}{2}-C_1\right)\bar{\rho}\tilde{R}_{ij}\frac{\partial\tilde{u}_i}{\partial x_j}-\left(\frac{3}{2}-C_2\right)\bar{\rho}\sqrt{K}\\ &-C_3\bar{\rho}S\frac{\partial\bar{u}_j}{\partial x_J}+C_s\frac{\partial}{\partial x_m}\left(\frac{S}{\sqrt{K}}\bar{\rho}\tilde{R}_{mn}\frac{\partial S}{\partial x_n}\right)\end{aligned}\tag{4}$$

2.1 Geometry

For this problem, we examine two parallel flowing fluid streams separated by a thin splitter plate. We examine multiple cases to compare to our model. These include shear only, buoyancy only, and shear and buoyancy combined with varying initial conditions. The mixing layer consists of two fluids with a combination of different velocities, thermodynamics states, or physical properties. The schematic of the buoyancy and shear mixing layer can be seen in Fig. 1, in which U is velocity, T is temperature, h is half the mixing width, and u is the average velocity . A picture of the actual experimental cases from Texas A&M University are shown in Fig. 2, where (a) Pure Rayleigh Taylor, (b) Pure Kelvin-Helmholtz, and (c) Combined Rayleigh-Taylor and Kelvin-Helmholtz instabilities are shown. [1]

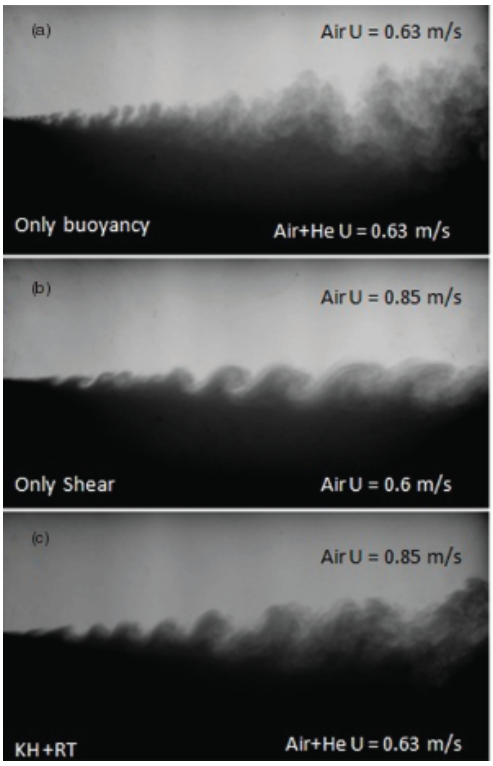


Figure 2: Experimental Cases (taken from [1])

2.2 Kelvin-Helmholtz and Rayleigh Taylor Instabilities

Rayleigh-Taylor Instability (RT) occurs when the interface between two fluids of different densities are accelerated such that $\bigtriangledown p \bullet \bigtriangledown \rho < 0$, where ρ is density and p is pressure. An important characteristic of RT is the Atwood number, At, which is a non-dimensional measure of the density difference between two fluids. Equation 5 shows the Atwood number equation where ρ_1 is the heavier fluid density and ρ_2 is the lighter fluid density.[1]

$$\mathrm{At} = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2} \tag{5}$$

Classically, small perturbations at the interface grow in size with time and begin interacting. In contrast to RTI, Kelvin-Helmholtz instability (KH) occurs when two fluids of different velocities interact at the interface. Like RT, this leads to turbulence. In time, the small perturbations at the interface grow and form into vortices. The relative strength of buoyancy to shear flow can be quantified by the Richardson number (Ri), which is shown in Equation 6, where g is gravity, h is mixing width, and ΔU is velocity difference. [1]

$$\mathrm{Ri} = \frac{-4gh\mathrm{At}}{(\Delta U)^2} \tag{6}$$

For this problem, several cases are run: RT(pure Rayleigh-Taylor) and KHRT1 and KHRT2 (Kelvin-Helmholtz and Rayleigh-Taylor combined).

3 Experimental Data and Setup

An experimental group at Texas A & M University provided data for comparison to the BHR-3 model. The BHR-3 model was initialized using using the initial conditions of the experimental data. A wind tunnel type facility was built at the university for this problem. The facility consists of two flow sections separated by a splitter plate. Each section is fashioned with flow straighteners and wire mesh screens to aid uniform

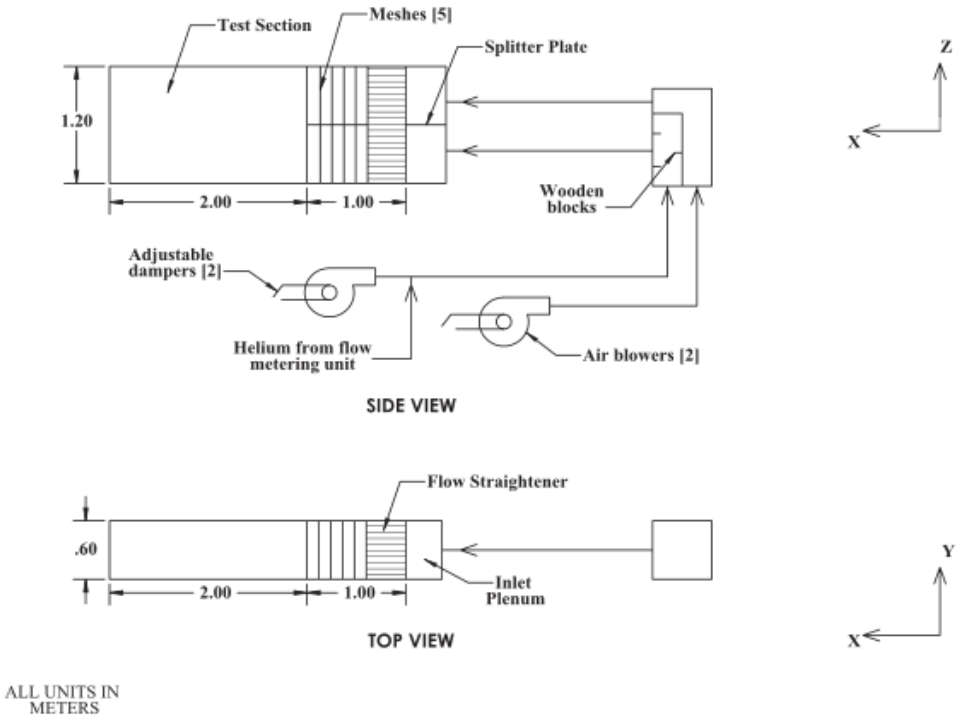


Figure 3: Experimental Setup (taken from [1])

Table 1: Stations

Case	Stations
RT	10, 40, 65, 100, 125, 160
KHRT1	10, 40 , 50, 65, 75, 100, 125, 150, 175
KHRT2	10, 50, 65, 75, 85, 100, 125, 150, 175

flow and reduce initial turbulence. Air is supplied to the top section, while an air and helium mixture are supplied to the bottom section. This creates an At of 0.035. The test section following the splitter plate is 2 meters long. Sensors are placed at certain positions along the test chamber for each case. The cases and stations are listed in Table 1. A schematic of the actual experimental set up is shown in Fig. 3

Using imaging and simultaneous hot wire and cold wire anemometry, measurements are made for mix widths, point-wise velocities, and densities. The specific experimental cases are shown in Table 2. The initial conditions taken from the data include R_{xy}, R_{yy}, k, a_x, a_y, b, and ϵ.

4 Analysis

It can be seen when comparing the BHR-3 model with the experimental data that the model is very sensitive to initial conditions. Initially, a parameter study comparing initial turbulent kinetic energy was conducted in order to see how the model behaves. Figures 4 and 5 show a comparison of normalized turbulent kinetic energy and Richardson number with varying initial turbulent kinetic energy. In early Richardson number, the flow converges to a specific trajectory. Prior to that, the response to the initial conditions can be seen.

A graph of the comparison of different initial condition profiles can be seen in Fig. 6, which shows now non-dimensional mixing width changes with non-dimensional distance. Data from [3] is compared to different models for initial conditions. "Parabolic" has peak values of K and ϵ, with a parabolic shape. "Peak" is

Table 2: Experimental Cases

Case	Bottom Stream Velocity, m/s (U_2)	Top Stream Velocity, m/s (U_1)	$U_{average}$, m/s	A_t	Maximum Ri number
RT	0.63	0.63	0.63	0.035	
KHRT1	0.63	0.86	0.75	0.035	-6.9
KHRT2	0.63	1.03	0.83	0.035	-1.8

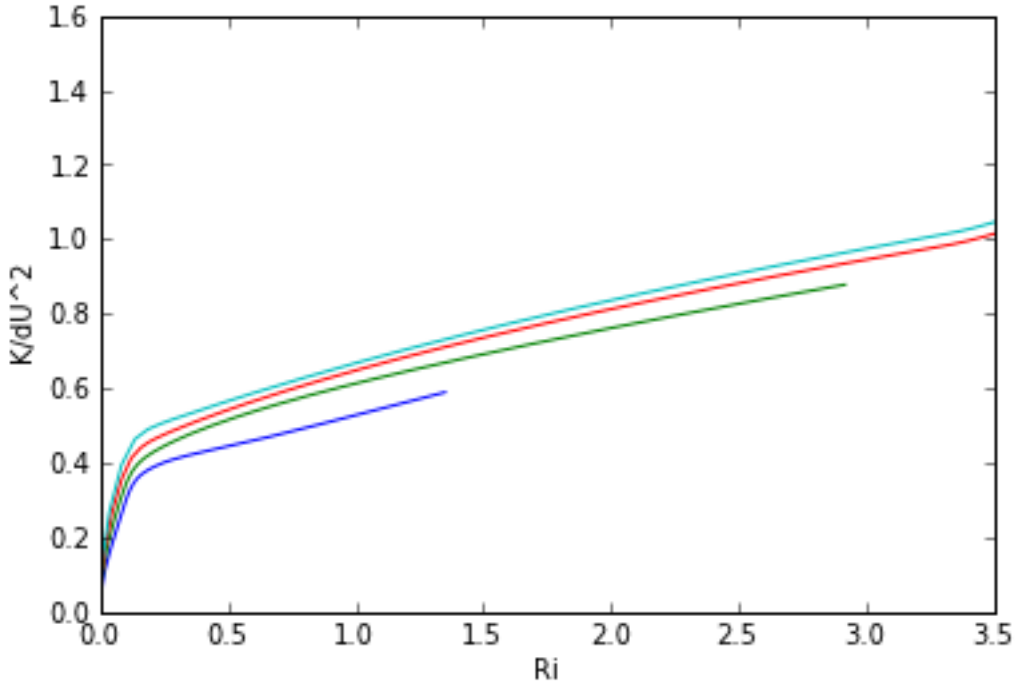


Figure 4: Parameter study KHRT1: varying initial turbulent kinetic energy

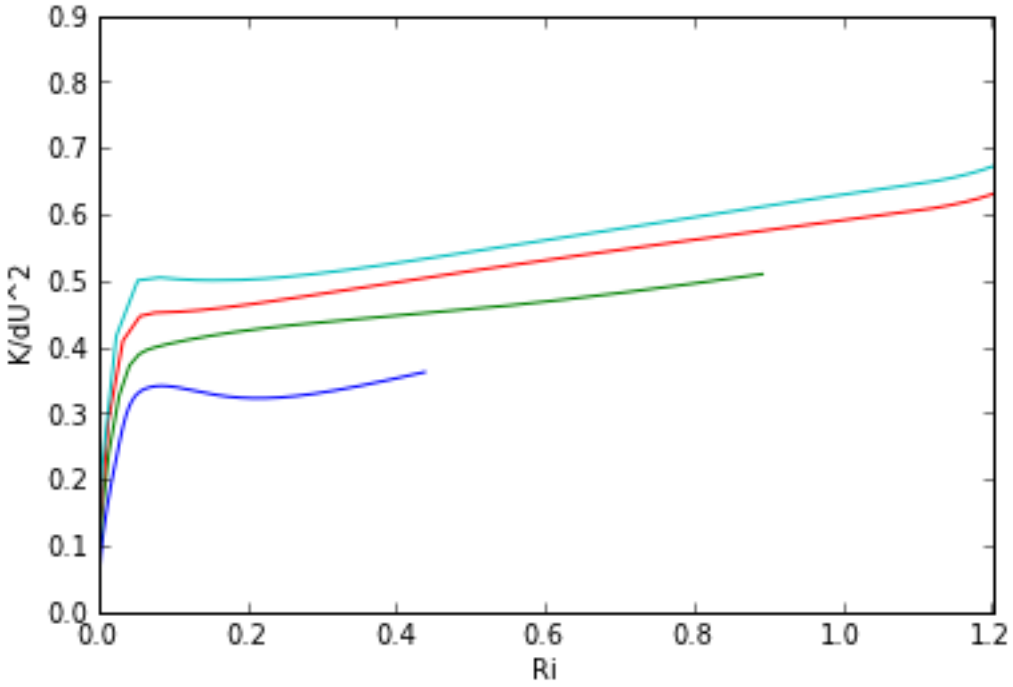


Figure 5: Parameter study KHRT2: varying initial turbulent kinetic energy

a uniform profile which uses the peak values of the "parabolic" IC. "Freestream" is a uniform profile with the freestream values of K and ϵ. "5%" assumes that the turbulent kinetic energy is five percent of U_{avg}^2, and ϵ is based on tunnel mesh length scale. It can be seen that "peak" and "parabolic" seem to fit the data the best, however, a slight change in the initial conditions can dramatically change how model behaves and fits the data. The initial conditions used for the BHR-3 model was a linear profile for velocity and density. A parabolic profile was chosen for all turbulent quantities because the experimental data shows an approximately parabolic fit.The initial mixing width was set based on experimental width data and the amplitude was based on experimental center line data. ϵ is scaled based on K and length scale, S, as seen in Equation 7. In [4], the coefficients were tuned to match DNS data. Initial S was tuned to match experimental data. For this project, that tuning was not done. Instead, when the initial conditions were set, there was no value for S, and so S was set to the layer thickness. This setting is most likely leads to inaccuracy, and an improved method for finding S is necessary.

$$\epsilon = \frac{K^{\frac{3}{2}}}{S} \tag{7}$$

Since the data and the model report different α, which is the dimensionless growth rate of the mixing layer in the self-similar regime, a direct comparison is difficult, hence the scaling issue. After coding the BHR-3 model equations, we could compare the experimental data of turbulent kinetic energy and mixing width height to the model. What can be seen from figures 7 and 8 is that we have not yet achieved a good enough agreement between the model and the data.

5 Conclusions and Future Work

Thus far, we have not achieved a good enough agreement between the experimental data from Texas A&M and the BHR-3 model. Possible explanations for this lack of agreement is that the scaling for ϵ is incorrect. This can be due to the lack of measured data for ϵ. There could also have been errors in post-processing the data. The BHR-3 model could need further calibration in the transient regime. Also, the initial condition

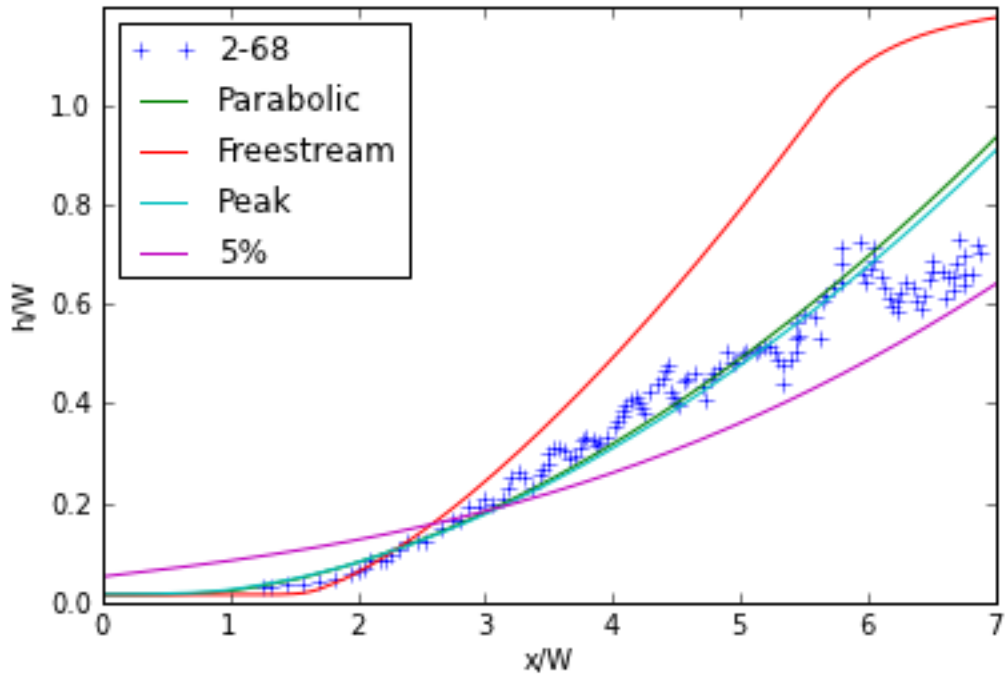


Figure 6: Comparison of Initial Conditions: Non-dimensional Distance vs. Mixing width height

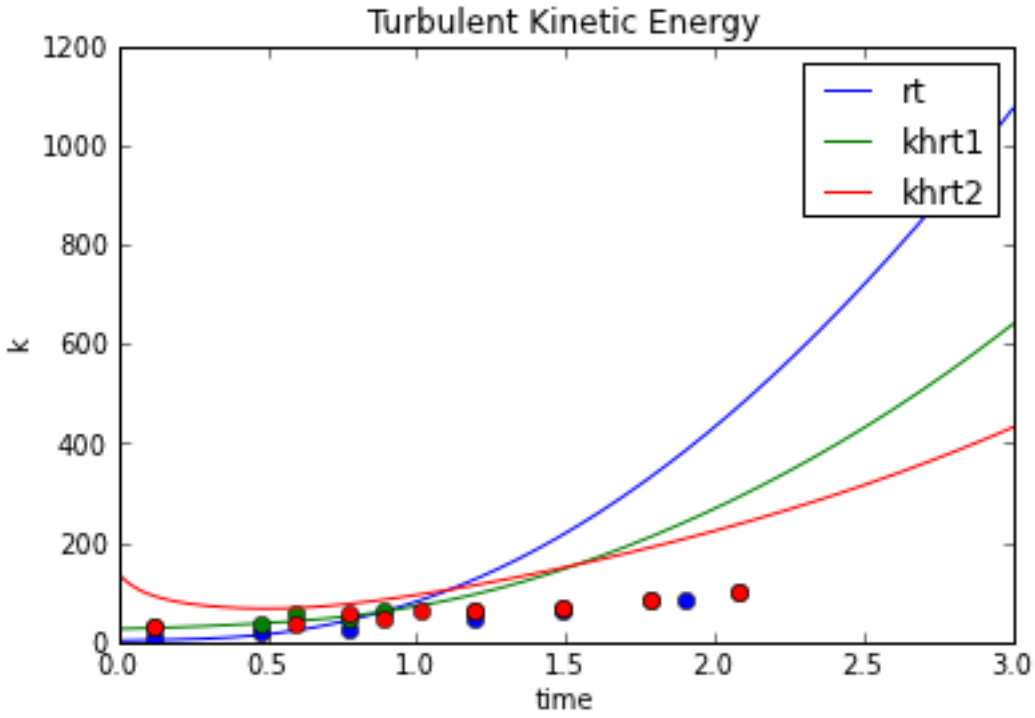


Figure 7: Turbulent Kinetic Energy

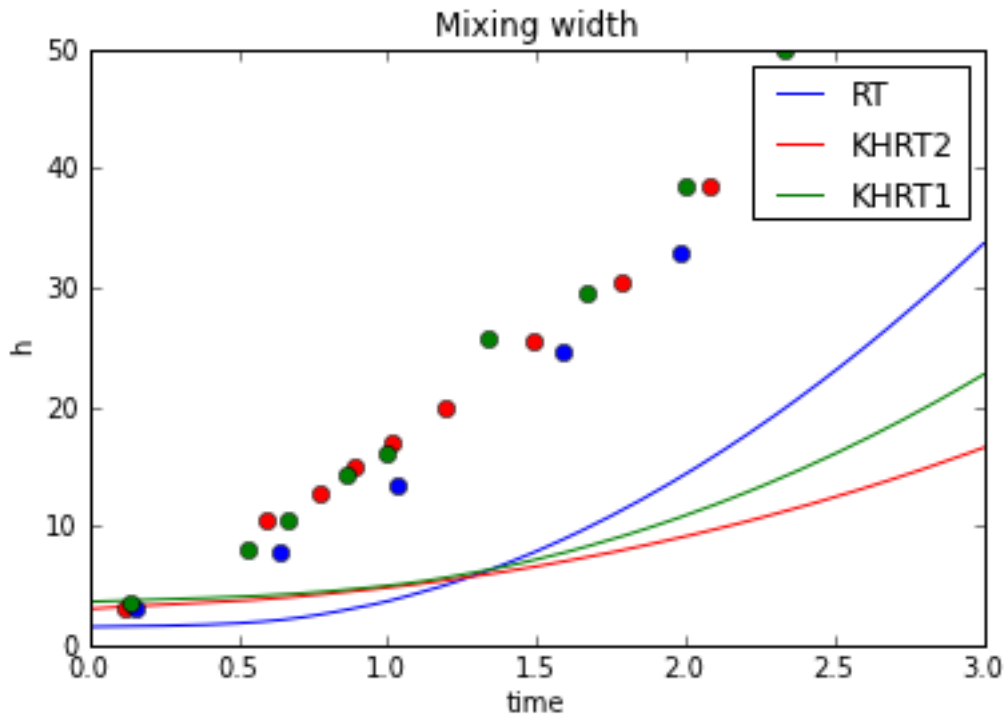


Figure 8: Mixing Width

profiles are based on center line data only. What can be concluded is that the model is much more sensitive to initial condition changes than was originally expected. Since more calibration is done in a self-similar regime, calibration for a transient problem is very difficult. The goals for the future work include resolving the lack of agreement, as well as obtaining complete profiles for experimental initial conditions. Additionally, an error analysis would be run after better agreement is achieved.

References

[1] Bhanesh Akula, Malcolm J. Andrews, Devesh Ranjan, *Effect of shear on Rayleigh-Taylor mixing at small Atwood number*. American Physical Society, PHYSICAL REVIEW E 87, 033013, 2013.

[2] Didier Besnard, Francis H. Harlow, Rick M. Rauenzahn, Charles Zemach, *Turbulence Transport Equations for Variable-Density Turbulence and Their Relationship to Two-Field Models*. Los Alamos National Laboratory, U.S Government Printing Office.

[3] D.M Snider and M. J. Andrews, *The simulation of mixing layers driven by compound buoyancy and shear*. ASME, Vol. 118, June 1996.

[4] John D. Schwarzkopf, Daniel Livescu, Robert A. Gore, Rick M. Rauenzahn, and J. Raymond Ristorcelli, *Application of a second-moment closure model to mixing processes involving multicomponent miscible fluids*. Los Alamos National Laboratory, Taylor & Francis, 2011.